Optimisation Globale basée sur l'Analyse d'Intervalles: Relaxation Affine et Limitation de la Mémoire

Jordan Ninin

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Institut National Polytechnique de Toulouse

Discipline ou spécialité : Informatique

Présentée et soutenue par

Jordan Ninin

Le 8 décembre 2010

Optimisation Globale basée sur l'Analyse d'Intervalles

Relaxation Affine et Limitation de la Mémoire

École doctorale

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche

Institut de Recherche en Informatique de Toulouse (IRIT)

Directeur de thèse

Frédéric Messine	Maître de Conférence HDR	INP - ENSEEIHT

Rapporteurs

Ralph Baker Kearfott	Professeur d'Université	University of Louisiana at Lafayette
Leo Liberti	Professeur d'Université	École Polytechnique

Examinateurs

Laurent Granvilliers	Professeur d'Université	Université de Nantes
Pierre Hansen	Professeur d'Université	HEC Montréal
Didier Henrion	Directeur de Recherche	LAAS-CNRS
Marcel Mongeau	Maître de Conférence HDR	Université Paul Sabatier

Remerciements

Que de chemin parcouru, que de gens passionnants rencontrés, que de lieux magnifiques visités et d'expériences enrichissantes partagées depuis le début de ce travail. Cette thèse est certes personnelle, mais elle n'aurait jamais été possible sans l'aide et le soutien de toutes les personnes qui m'ont entouré.

Tout d'abord, je tiens à remercier Frédéric Messine pour son encadrement, sa gentillesse et son amitié. Merci pour les centaines de litres de café bu tout en discutant de recherche durant des centaines d'heures. C'est grâce à cette rencontre que j'ai su que je voulais faire de la recherche. Il a su me faire profiter de toute son expérience et de son savoir avec générosité et m'a appris les rudiments de ce passionnant métier de chercheur. J'ai eu grand plaisir à travailler à ses côtés et j'espère poursuivre ma carrière dans le même genre d'ambiance.

Merci à Pierre Hansen d'avoir accepté de présider mon jury de thèse et surtout pour m'avoir fait confiance dès le début de mon Master. Son expérience, son érudition et sa sagesse m'ont permis de progresser grandement. Et chacune de nos rencontres a été pour moi extrêmement enrichissante.

Merci à Leo Liberti d'avoir accepté d'être rapporteur de ma thèse et pour les discussions, toujours fructueuses, que nous avons eues. Son dynamisme et sa confiance m'ont toujours poussé à échanger et à communiquer davantage sur mes idées aux quatre coins du monde.

Je tiens également à remercier Ralph Baker Kearfott pour avoir accepté d'être rapporteur de ma thèse, et d'avoir fait le déplacement pour la soutenance. Et je suis certain que de cette première rencontre débutera une collaboration fructueuse.

Merci encore à Laurent Granvilliers, Didier Henrion et Marcel Mongeau d'avoir accepté de faire partie de mon jury de thèse. Merci également pour vos conseils et vos remarques qui m'ont permis d'améliorer et de clarifier mon travail. Et j'ai hâte de pouvoir continuer à travailler avec vous.

Merci à tous les membres de l'équipe APO qui m'ont accueilli au sein de l'IRIT. Merci à mes deux coreligionnaires Olivier et Sandrine pour leur soutien et les centaines d'heures passées ensemble dans ce bureau avec une fenêtre donnant sur une fenêtre donnant elle-même sur un mur, à la recherche d'une lueur de soleil et d'un peu d'air frais. Merci à toute l'équipe GREM3 du LAPLACE pour l'accueil toujours chaleureux qui m'a été fait et les applications toujours étonnantes et surprenantes sur lesquelles ils m'ont proposé de travailler.

Je tiens également à remercier tous mes amis, Olivier, les anciens de TVn7, les anciens de l'N7, les amis du groupe nano, les amis du V&B, les copains de Reims, et tous ceux avec qui j'ai passé des moments inoubliables.

Merci à mes parents, mon frère et mes grands-parents pour la confiance sans faille qu'ils me témoignent, pour les encouragements à aller toujours plus loin et pour cette curiosité et cette soif d'apprendre qu'ils ont su m'inculquer.

Enfin, il est impossible pour moi de passer sous silence l'influence positive qu'a eue pour moi Laurène, avec qui je partage chaque instant depuis trois ans. Son amour a été pour moi une source inépuisable d'énergie, de force et de stabilité. Son point de vue a toujours été essentiel et m'a permis de rendre mon travail plus lisible et mieux construit. Elle n'imagine pas à quel point son aide a été précieuse pour moi. J'espère simplement pouvoir un jour lui rendre autant qu'elle m'a donné et continue de me donner.

Résumé

Depuis une vingtaine d'années, la résolution de problèmes d'optimisation globale non convexes avec contraintes a connu un formidable essor. Les algorithmes de branch and bound basée sur l'analyse d'intervalles ont su trouver leur place, car ils ont l'avantage de prouver l'optimalité de la solution de façon déterministe, avec un niveau de certitude pouvant aller jusqu'à la précision machine. Cependant, la complexité exponentielle en temps et en mémoire de ces algorithmes induit une limite intrinsèque, c'est pourquoi il est toujours nécessaire d'améliorer les techniques actuelles.

• Dans cette thèse, nous avons développé de nouvelles arithmétiques basées sur l'arithmétique d'intervalles et l'arithmétique affine, afin de calculer des minorants et des majorants de meilleure qualité de fonctions explicites sur un intervalle.

• Nous avons ensuite développé une nouvelle méthode automatique de construction de relaxations linéaires. Cette construction est basée sur l'arithmétique affine et procède par surcharge des opérateurs. Les programmes linéaires ainsi générés ont exactement le même nombre de variables et de contraintes d'inégalité que les problèmes originaux, les contraintes d'égalité étant remplacées par deux inégalités. Cette nouvelle procédure permet de calculer des minorants fiables et des certificats d'infaisabilité pour chaque sous-domaine à chaque itération de notre algorithme de branch and bound par intervalles. De nombreux tests numériques issus du site COCONUT viennent confirmer l'efficacité de cette approche.

• Un autre aspect de cette thèse a été l'étude d'une extension de ce type d'algorithmes en introduisant une limite sur mémoire disponible. L'idée principale de cette approche est de proposer un processus inverse de l'optimisation par le biais d'un principe métaheuristique : plutôt que d'améliorer des solutions locales à l'aide de métaheuristiques telles que les algorithmes Taboo ou VNS, nous partons d'une méthode exacte et nous la modifions en une heuristique. De cette façon, la qualité de la solution trouvée peut être évaluée. Une étude de la complexité de ce principe métaheuristique a également été effectuée.

• Enfin, pour finir l'étude, nous avons appliqué notre algorithme à la résolution de problème en géométrie plane, ainsi qu'à la résolution d'un problème de dimensionnement de moteur électrique. Les résultats obtenus ont permis de confirmer l'intérêt de ce type d'algorithme, en résolvant des problèmes ouverts sur les polygones convexes et proposant des structures innovantes en génie électrique.

Mots Clés : Recherche Opérationnelle, Optimisation Globale, Branch and Bound, Analyse d'Intervalles, Arithmétique Affine, Calcul Robuste, Relaxation Linéaire.

Abstract

Since about thirty years, interval Branch and Bound algorithms are increasingly used to solve constrained global optimization problems in a deterministic way. Such algorithms are *reliable*, i.e., they provide an optimal solution and its value with guaranteed bounds on the error, or a proof that the problem under study is infeasible. Other approaches to global optimization, while useful and often less time-consuming than interval methods, do not provide such a guarantee. However, the exponential complexity in time and memory of interval Branch and Bound algorithms implies a limitation, so it is always necessary to improve these methods.

• In this thesis, we have developed new arithmetics based on interval arithmetic and affine arithmetic, to compute better lower and upper bounds of a factorable function over an interval.

• An automatic method for constructing linear relaxations of constrained global optimization problems is proposed. Such a construction is based on affine and interval arithmetics and uses operator overloading. These linear programs have exactly the same numbers of variables and of inequality constraints as the given problems. Each equality constraint is replaced by two inequalities. This new procedure for computing reliable bounds and certificates of infeasibility is inserted into a classical interval Branch and Bound algorithm. Extensive computation experiments, made on a sample of test problems from the COCONUT database, prove its effectiveness.

• Another aspect is the study of an extension of such a global optimization code by limiting the available memory. The main idea of this new kind of metaheuristique is to propose a reverse process of optimization via heuristics : rather than improving local solutions by using metaheuristics such as Taboo or VNS, we begin with an exact method and we modify it into a heuristic one. In such a way, the quality of the solution could be evaluated. Moreover, a study of the complexity of this metaheurisque has been done.

• Finally, we applied our algorithm to solve open problem in geometry, and to solve a design problem of an electric motor. The results have confirmed the usefulness of this kind of algorithms, solving open problems on convex polygons and offering innovative structures in electrical engineering.

Keywords : Operations Research, Global Optimization, Branch and Bound, Interval Analysis, Affine Arithmetic, Reliable Computation, Linear Relaxation.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Introduction

Nous faisons tous de l'optimisation. Dans notre vie quotidienne, nous cherchons à optimiser notre temps de travail, nos espaces de rangement, ou encore le trajet que nous aurons à parcourir pour nous rendre quelque part, etc. Nous recherchons tous une meilleure solution aux problèmes qui jalonnent notre existence. De manière générale, l'optimisation va donc consister à trouver cette meilleure solution.

Comme nous le rappelle l'adage populaire selon lequel : "les mathématiques permettent de mettre le monde en équation", il peut être tracé un parallèle entre l'optimisation quotidienne et celle, plus technique que l'on retrouve en science. En mathématique, la meilleure solution se recherche au sein d'un domaine initial. Cette solution est souvent soumise à des contraintes qui correspondent à des obligations ou des souhaits à respecter. Le critère permettant de distinguer qu'une solution est la meilleure s'appelle la *fonction objectif*. L'optimisation mathématique va consister à rechercher dans le domaine initial une solution qui maximise ou minimise une fonction objectif tout en respectant des contraintes.

Pour un domaine continu, on distingue classiquement deux types d'optimisation :

- **L'optimisation locale** recherche une solution qui est la meilleure localement, c'est-à-dire que dans son voisinage aucune solution n'est meilleure qu'elle. Cette solution est appelée un optimum local.
- **L'optimisation globale** recherche quant à elle la meilleure solution du domaine en entier, c'est-à-dire que dans tout le domaine il n'existe aucune solution qui lui soit meilleure tout en respectant les contraintes. Cette solution est appelée l'optimum global.

Par définition, l'optimum global est aussi une solution locale. En revanche, il est bien plus épineux de trouver l'optimum global, car lorsque l'on pense avoir trouvé cet optimum, sa démonstration se révèle bien souvent particulièrement ardue.

L'intérêt de l'optimisation globale par rapport à l'optimisation locale est patent. Elle garantit en effet que personne ne peut avoir une solution meilleure que celle trouvée. Or, pour une entreprise, cette information a son importance, car la différence entre la solution globale et une solution locale est bien souvent significative. Mais l'intérêt n'est pas que compétitif. Dans de nombreux problèmes, l'optimum global est la seule solution mathématique correspondant à une réalité physique. C'est ce qu'illustre par exemple la recherche de la quantité de chaque élément présent dans un mélange chimique à l'équilibre.

De nos jours, afin de résoudre des problèmes d'optimisation globale avec contraintes, de nombreuses stratégies algorithmiques s'avèrent disponibles. Pour guider le choix de la meilleure stratégie à utiliser, il est nécessaire de regarder : (i) la taille du problème, (ii) les propriétés de la fonction objectif et des contraintes (continuité, différentiabilité, linéarité, convexité,...), (iii) ainsi que le temps disponible pour résoudre le problème. Voici une liste non exhaustive des différentes approches :

- **Les méthodes multistart**
  Ces méthodes sont entièrement basées sur les techniques de recherche d'optima locaux. Le principe consiste à effectuer plusieurs recherches locales dans des parties du domaine déterminées aléatoirement. L'avantage est que ces méthodes sont généralement faciles à programmer et la solution trouvée est au moins un optimum local. Par ailleurs, si le problème considéré a 10 optima locaux, en effectuant 1000 optimisations locales dans des parties aléatoires du domaine, il y a de grandes chances de trouver l'optimum global. Mais, si le problème considéré a 10000 optima locaux, en effectuant 1000 optimisations locales, ces chances sont très faibles. Or, il est généralement impossible a priori de déterminer combien d'optima locaux possèdent le problème.
- **Les algorithmes évolutionnaires**
  les algorithmes les plus connus de cette classe de méthode sont : les algorithmes génétiques ou encore les colonies de fourmis, etc. Ils consistent à faire évoluer une population de solutions en effectuant des croisements entre elles ainsi que des mutations, et ce, tout en ne gardant que les meilleurs éléments d'une génération à l'autre. Malheureusement (ou heureusement), ce principe d'évolution est beaucoup plus efficace dans la nature qu'en mathématique. Car, pour que ces techniques soient performantes, il est nécessaire d'avoir une bonne connaissance du problème à résoudre, afin d'ajuster les paramètres, les règles de croisements ainsi que les règles de mutations.
- **Les méthodes métaheuristiques**
  Tout comme dans les méthodes multistart, ces techniques sont basées sur un algorithme d'optimisation locale, mais dans cette stratégie, l'algorithme utilisé est combiné avec une stratégie pour explorer plus largement le domaine de recherche. En d'autres termes, l'exploration du domaine initial se réalise de façon moins aléatoire, en privilégiant ou interdisant certaines zones. Les deux métaheuristiques les plus connues sont la recherche Tabou et la recherche à voisinage variable (VNS).
- **Les méthodes déterministes globales**
  Dans ce type de méthode, l'aléatoire n'intervient pas, c'est-à-dire que pour résoudre un problème, l'algorithme se comportera toujours de la même façon et donnera toujours la même réponse. Ces algorithmes peuvent se classer en fonction du type de problèmes pouvant être résolu : les programmes linéaires, les problèmes convexes, quadratiques, polynomiaux ou plus généraux.

  Ces techniques ont généralement l'avantage de ne pas nécessiter de point de départ. Mais avant tout elles fournissent une réponse déterminante sur la qua-

lité des solutions trouvées : l'optimum est-il local ou global ? Quel est le degré de certitude ? etc. Cette précision a une importance significative, car il est souvent beaucoup moins coûteux de trouver une solution que de prouver qu'il s'agit bien de l'optimum global.

Dans notre étude, nous nous sommes concentrés sur la résolution de *problèmes explicites*, c'est-à-dire des problèmes dans lesquels toutes les expressions des équations de la fonction objectif et des contraintes sont connues de façon explicite. Nous exclurons donc les fonctions de type *boîte noire*. Néanmoins, nous n'imposerons que peu de limites quant à la complexité des équations : celles-ci pouvant non seulement être linéaires, quadratiques, polynomiales, convexes, non convexes, etc., mais aussi utiliser les fonctions : exponentielles, logarithmique, la racine carrée, la division, etc. Ainsi, le nombre de variables est souvent réduit, mais la complexité du problème provient de la nature même des équations.

Les algorithmes que nous allons étudier font parties des méthodes déterministes globales permettant de résoudre des problèmes généraux d'optimisation. Par ailleurs, ils permettent de fournir le plus haut degré de certitude concernant l'optimalité globale des solutions trouvées ; ces algorithmes vont même jusqu'à prendre en compte les erreurs numériques intrinsèques au calcul sur ordinateur. Ceux-ci s'appellent : **Algorithmes de Branch and Bound par Intervalles**. Ils peuvent théoriquement résoudre n'importe quel problème, pour peu que l'on dispose de suffisamment de temps ainsi que d'une mémoire infinie sur l'ordinateur. Bien entendu cette hypothèse n'étant pas plausible, il est donc nécessaire de développer des techniques afin de limiter la mémoire nécessaire et d'accélérer la convergence de ce type d'algorithme.

Le premier chapitre de notre étude commence par un rappel des notions élémentaires de l'analyse d'intervalles ainsi que des notations et définitions qui en découlent, afin d'établir clairement les bases sur lesquelles repose notre approche. Nous y détaillerons le principe de l'algorithme de Branch and Bound par intervalles, en mettant en évidence son caractère modulaire. Quelques techniques d'accélération précédemment étudiées y seront également rappelées.

Le deuxième chapitre est consacré quant à lui à l'étude des extensions de l'arithmétique affine. Dans un premier temps, l'arithmétique affine standard introduite par Comba et Stolfi y est présentée, ainsi que les travaux de Messine sur ces extensions. Une nouvelle extension y est décrite basée sur le même principe que celles de Messine en modifiant la variable de la forme affine. Nous proposons également une nouvelle méthode de combinaison de l'arithmétique d'intervalles et de l'arithmétique affine. Celle-ci permet d'obtenir des encadrements de fonction de meilleure qualité, en combinant les avantages de chacune des deux arithmétiques. Puis, une étude sur l'implémentation de la robustesse de l'arithmétique est effectuée. Trois méthodes y sont détaillées et comparées.

Dans le troisième chapitre, nous proposons une méthode automatique de construction de relaxation linéaire de problèmes d'optimisation avec contraintes. Cette

construction est basée sur l'arithmétique affine et l'arithmétique d'intervalles, en utilisant la surcharge d'opérateurs. Le programme linéaire généré a exactement le même nombre de variables et de contraintes d'égalité que le problème original. Chaque contrainte d'égalité est remplacée par deux inégalités. Ce nouveau procédé pour calculer des bornes fiables et des certificats d'infaisabilité est inséré dans notre algorithme de Branch and Bound par intervalles. De nombreuses expériences numériques tirées de la littérature sont effectuées, venant confirmer l'efficacité de cette nouvelle technique d'accélération.

Dans le quatrième chapitre, nous nous intéressons au problème lié à la complexité exponentielle intrinsèque des algorithmes de Branch and Bound par intervalles. L'idée que nous proposons consiste à limiter la mémoire disponible pour l'exécution, afin de trouver efficacement des solutions réalisables. De cette façon, nous introduisons un principe métaheuristique permettant de développer de nouveaux algorithmes heuristiques d'optimisation globale basée sur une méthode exacte. Moyennant une petite hypothèse sur le tri en largeur d'abord de la structure de données, nous montrons que la complexité en temps d'un tel algorithme métaheuristique devient polynomiale. Afin de valider notre approche métaheuristique, des expériences numériques sont réalisées sur la résolution de problèmes d'optimisation globale avec contraintes. L'heuristique utilisée permet de certifier l'encadrement de la valeur du minimum global. L'objectif premier de cette heuristique n'est pas de résoudre des problèmes ou de trouver une meilleure solution, mais de savoir quelle est la meilleure précision pouvant être garantie de manière fiable avec la mémoire vive disponible.

Le dernier chapitre conclura notre étude avec les deux applications suivantes :

- La première concerne la résolution des trois questions distinctes de maximisation du périmètre, du diamètre et de l'aire d'un polygone équilatéral convexe de largeur unitaire. L'énoncé d'une conjecture sur la question de la maximisation de la somme des distances d'un polygone équilatéral convexe de largeur unitaire est également prouvé numériquement pour les pentagones, heptagones et ennéagones. La solution à ces quatre problèmes tend trivialement vers l'infini lorsque le nombre de côtés est pair. Nous montrons que lorsque ce nombre est impair, la solution optimale à ces problèmes est identique, et est arbitrairement proche d'un trapèze.
- La deuxième application consiste à résoudre un problème de dimensionnement de moteurs électriques. Le modèle utilisé y est entièrement détaillé, en spécifiant notamment les variables, les contraintes, les fonctions objectifs et la formulation générale utilisée. Sa résolution est effectuée en utilisant notre algorithme de Branch and Bound par intervalles.

Chapitre 1

Technique de Branch&Bound par Intervalles

1.1 Analyse d'intervalles

L'analyse d'intervalles est née dans les années 1960 avec l'avènement des premiers ordinateurs. En effet, au début de l'invention des calculs par ordinateur, les formats pour coder les nombres flottants n'étaient pas encore standardisés et des erreurs numériques pouvaient s'accumuler et aboutir à des résultats aberrants. L'exemple 1.1.1 montre qu'aujourd'hui encore, malgré la standardisation IEEE-754 du codage des nombres flottants [2], on n'est jamais à l'abri des erreurs d'arrondi numérique. $\mathbb{R}$ étant infini et indénombrable, il est impossible de représenter exactement tous les nombres. L'analyse d'intervalles a donc été développée à ses débuts pour contourner ce problème.

Exemple 1.1.1 *Considérons l'équation suivante :*

$$f(x,y) = 33.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y}.$$

En l'évaluant au point $x = 77617$ *et* $y = 33096$*, en Fortran double précision,*

$$\textit{on obtient :} \quad f(77617, 33096) = 1.1726039400531...$$

$$\textit{Alors que le vrai résultat est :} \quad f(77617, 33096) = -\frac{54767}{66192} = -0.8273960599.$$

Comme on peut le constater, le vrai résultat n'a aucun rapport avec la valeur retournée par l'ordinateur, même le signe n'est pas correct.

Le premier livre sur ce domaine est celui de Moore en 1966 [78]. Ce n'est pas historiquement le premier travail sur ce sujet, mais c'est à la suite de ce livre que plus de 1000 articles furent écrits, ce qui donna un formidable élan à l'analyse d'intervalles.

L'idée de l'analyse d'intervalles est de représenter tous les nombres réels par deux nombres flottants qui l'encadrent.

Exemple 1.1.2 *Pour représenter* $1/3$ *sur machine, on utilise un intervalle contenant la valeur réelle :*

$1/3 \rightarrow [0.33333331, 0.33333335]$ *avec un codage simple précision,*
$1/3 \rightarrow [0.33333333333333331, 0.33333333333333338]$ *en double précision.*

1.1.1 Arithmétique d'intervalles

Définissons un intervalle $\mathbf{x}$ comme étant une paire ordonnée de nombres réels $[\underline{\mathbf{x}}, \overline{\mathbf{x}}]$, avec $\underline{\mathbf{x}} \leq \overline{\mathbf{x}}$. $\underline{\mathbf{x}}$ représente la borne inférieure de $\mathbf{x}$ et $\overline{\mathbf{x}}$ sa borne supérieure.

$$\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}] = \{x \in \mathbb{R} | \underline{\mathbf{x}} \leq x \leq \overline{\mathbf{x}}\}.$$

Notons $\mathbb{I}$, l'ensemble des intervalles.

$$\mathbb{I} = \{\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}] / \underline{\mathbf{x}}, \overline{\mathbf{x}} \in \mathbb{R} \text{ et } \underline{\mathbf{x}} \leq \overline{\mathbf{x}}\}.$$

Soit $\mathbf{x}$ et $\mathbf{y}$ deux intervalles appartenant à $\mathbb{I}$. Nous allons maintenant redéfinir dans $\mathbb{I}$ les opérateurs usuels de l'arithmétique euclidienne. La plupart des opérateurs et des relations gardent leur définition usuelle sur $\mathbb{I}$, en revanche leurs propriétés sont souvent affaiblies.

Par exemple, la relation d'ordre $<$ est une relation d'ordre partiel sur $\mathbb{I}$:

$$\mathbf{x} < \mathbf{y} \text{ si et seulement si } \overline{\mathbf{x}} < \underline{\mathbf{y}}.$$

Les opérateurs sur les ensembles gardent la même définition pour des intervalles. Ainsi, l'union et l'intersection de deux intervalles donnent :

$$\begin{array}{rl} \mathbf{x} \cup \mathbf{y} = & [\min(\underline{\mathbf{x}}, \underline{\mathbf{y}}), \max(\overline{\mathbf{x}}, \overline{\mathbf{y}})], \\ \mathbf{x} \cap \mathbf{y} = & \left\{ \begin{array}{cl} \emptyset & \text{si } \overline{\mathbf{x}} < \underline{\mathbf{y}} \text{ ou } \overline{\mathbf{y}} < \underline{\mathbf{x}}, \\ [\max(\underline{\mathbf{x}}, \underline{\mathbf{y}}), \min(\overline{\mathbf{x}}, \overline{\mathbf{y}})] & \text{sinon.} \end{array} \right. \end{array}$$

Définissons $w(\mathbf{x})$ comme étant la largeur de l'intervalle $\mathbf{x}$, $w(\mathbf{x}) = \overline{\mathbf{x}} - \underline{\mathbf{x}}$.
Si $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathbb{I}^n$, alors $w(\mathbf{X}) = \max_{i \in \{1,\ldots,n\}} w(\mathbf{x}_i)$.
Notons $mid(\mathbf{x})$, le milieu de l'intervalle $\mathbf{x}$: $mid(\mathbf{x}) = \frac{\underline{\mathbf{x}} + \overline{\mathbf{x}}}{2}$.
Si $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathbb{I}^n$, alors $mid(\mathbf{X}) = (\frac{\underline{\mathbf{x}_1} + \overline{\mathbf{x}_1}}{2}, \frac{\underline{\mathbf{x}_2} + \overline{\mathbf{x}_2}}{2}, \ldots, \frac{\underline{\mathbf{x}_n} + \overline{\mathbf{x}_n}}{2})$.

Les opérateurs usuels de l'arithmétique d'intervalles sont définis de telle sorte que l'intervalle résultant soit le plus petit intervalle contenant tous les points images des éléments des intervalles de départ. Ainsi, la formule générale des opérateurs unaires et binaires devient :

$$\begin{array}{rll} \mathbf{x} \text{ op } \mathbf{y} = & \{x \text{ op } y \mid x \in \mathbf{x} \text{ et } y \in \mathbf{y}\} & \text{avec op } \in \{+, -, \times, \div\}. \\ u(\mathbf{x}) = & \{u(x) | x \in \mathbf{x}\} & \text{avec } u \in \{\sqrt{}, \log, \exp, abs, \ldots\}. \end{array}$$

On obtient les formules suivantes pour les opérateurs binaires de base :

$$\begin{cases} \mathbf{x}+\mathbf{y} = & [\underline{\mathbf{x}}+\underline{\mathbf{y}}, \overline{\mathbf{x}}+\overline{\mathbf{y}}], \\ \mathbf{x}-\mathbf{y} = & [\underline{\mathbf{x}}-\overline{\mathbf{y}}, \overline{\mathbf{x}}-\underline{\mathbf{y}}], \\ \mathbf{x}\times\mathbf{y} = & \left[\min\{\underline{\mathbf{x}}\times\underline{\mathbf{y}}, \underline{\mathbf{x}}\times\overline{\mathbf{y}}, \overline{\mathbf{x}}\times\underline{\mathbf{y}}, \overline{\mathbf{x}}\times\overline{\mathbf{y}}\}, \right. \\ & \left. \max\{\underline{\mathbf{x}}\times\underline{\mathbf{y}}, \underline{\mathbf{x}}\times\overline{\mathbf{y}}, \overline{\mathbf{x}}\times\underline{\mathbf{y}}, \overline{\mathbf{x}}\times\overline{\mathbf{y}}\}\right], \\ \mathbf{x}\div\mathbf{y} = & \mathbf{x}\times\left[\frac{1}{\overline{\mathbf{y}}}, \frac{1}{\underline{\mathbf{y}}}\right] \text{ si } 0\notin\mathbf{y}. \end{cases}$$

Il est aussi très simple d'étendre ce principe aux fonctions unaires. Voici quelques exemples d'opérateurs usuels :

$$\begin{array}{rll} \sqrt{\mathbf{x}} = & [\sqrt{\underline{\mathbf{x}}}, \sqrt{\overline{\mathbf{x}}}], & \text{pour } \underline{\mathbf{x}} \geq 0, \\ \log(\mathbf{x}) = & [\log(\underline{\mathbf{x}}), \log(\overline{\mathbf{x}})], & \text{pour } \underline{\mathbf{x}} > 0, \\ \exp(\mathbf{x}) = & [\exp(\underline{\mathbf{x}}), \exp(\overline{\mathbf{x}})], & \\ abs(\mathbf{x}) = & \begin{cases} [abs(\underline{\mathbf{x}}), abs(\overline{\mathbf{x}})] & \text{si } \underline{\mathbf{x}} > 0, \\ [abs(\overline{\mathbf{x}}), abs(\underline{\mathbf{x}})] & \text{si } \overline{\mathbf{x}} < 0, \\ [0, \max(abs(\underline{\mathbf{x}}), abs(\overline{\mathbf{x}}))] & \text{si } 0 \in \mathbf{x}. \end{cases} & \end{array}$$

De nombreuses extensions furent proposées notamment pour inclure les intervalles infinis, ou par exemple pour gérer la division par un intervalle contenant 0 : cette arithmétique appelée *l'arithmétique d'intervalles étendue* a été découverte par Kahan et Hanson de façon indépendante [45, 50] et popularisée par Hansen [42].

Concernant l'implémentation de l'arithmétique d'intervalles sur machine, il est bien entendu impossible d'utiliser des nombres réels, une version n'utilisant que des nombres flottants est donc nécessaire. Cette arithmétique s'appelle *l'arithmétique d'intervalles arrondie* [79]. Elle consiste à approximer les nombres réels par le nombre flottant le plus proche de telle sorte que l'intervalle réel soit inclus dans l'intervalle arrondi. Soit $\mathbf{x} \in \mathbb{I}$, alors $\underline{\mathbf{x}}$ sera arrondi au nombre flottant inférieur le plus proche et $\overline{\mathbf{x}}$ sera arrondi au nombre flottant supérieur le plus proche. Une étude pour créer le standard IEEE-P1788 définissant cette arithmétique est actuellement en cours [3]. Mais, il existe déjà de nombreuses implémentations performantes de cette arithmétique, dont voici une liste non exhaustive dans différents langages de programmation :

- INTLIB en Fortran 77 et Fortran 90 [52]
- INTLAB en Matlab [94],
- PROFIL/BIAS en C [57],
- une librairie directement incluse dans le compilateur SUN f90 [48].

Ces différentes librairies permettent d'utiliser le type *interval* et de le manipuler comme tous les autres types. Les opérateurs et les fonctions usuelles sont simplement redéfinis par surcharge d'opérateur [76].

Remarque 1.1.3 *Tout au long de notre étude, nous avons réalisé tous nos tests numériques en programmant les arithmétiques et les algorithmes en Fortran 90/95. Nous avons donc utilisé la librairie codant l'arithmétique d'intervalles qui est intégrée*

dans le compilateur f90 *de SUN. Cette librairie s'active avec l'option* -xia *et permet d'utiliser directement le type* interval. *Pour les détails des algorithmes implémentés dans cette librairie voir [48].*

1.1.2 Extension naturelle et Propriétés

En pratique l'arithmétique d'intervalles est utilisée pour calculer des minorants et des majorants de fonction de $\mathbb{R}^n$ dans $\mathbb{R}$. Pour cela, introduisons d'abord quelques définitions.

Il existe des définitions plus précises d'une *fonction explicite* (*factorable fonctions* en anglais) et d'un *problème explicite* (*factorable program* en anglais). Mais, par souci de concision, nous avons préféré utiliser des définitions plus intuitives.

Définition 1.1.4 *Soit $f : \mathbb{R}^n \to \mathbb{R}$, f possède une* expression explicite *si l'expression analytique de f est connue de façon explicite, c'est-à-dire qu'elle peut s'écrire en n'utilisant que des variables, des fonctions et des opérateurs élémentaires tels que $+$, $-$, $\times$, $/$, $\sqrt{}$,* log, exp, abs, cos, sin, arccos. *On dit que f est une* fonction explicite.

Définition 1.1.5 *Un* problème explicite *est un problème dans lequel toutes les fonctions possèdent des expressions explicites et les contraintes n'utilisent que les relations standards ($\leq$, $<$, $\geq$, $>$, $=$).*

Définition 1.1.6 *Soit $f : \boldsymbol{X} \subset \mathbb{R}^n \to \mathbb{R}$. Une* fonction d'inclusion *de f est une fonction retournant un minorant et un majorant de f sur un intervalle donné. On la note, de manière générale, $F : \boldsymbol{X} \cap \mathbb{I}^n \to \mathbb{I}$. Ainsi :*

$$\forall z \in \mathbf{Z} \subseteq \boldsymbol{X}, f(z) \in F(\mathbf{Z}).$$

Définition 1.1.7 *Soit $f : \boldsymbol{X} \subset \mathbb{R}^n \to \mathbb{R}$. Une* extension naturelle *aux intervalles d'une fonction est la réécriture de f en remplaçant toutes les occurrences d'une variable par l'intervalle correspondant et les opérateurs classiques par leur équivalent en arithmétique d'intervalles. Elle est notée $EN_f(\boldsymbol{X})$.*

Toutes les *fonctions explicites* possèdent une *extension naturelle*. Mais, l'évaluation de celle-ci n'est pas unique. En effet, certaines fonctions peuvent s'écrire sous forme développée ou factorisée, il existe donc une extension naturelle pour chaque façon d'écrire la fonction.

Bien entendu, les *fonctions explicites* ne sont pas les seules à posséder une fonction d'inclusion. Il est par exemple possible d'écrire des fonctions d'inclusion de fonction comportant des $if..then...else$.

Proposition 1.1.8 *L'*extension naturelle *aux intervalles d'une fonction est une* fonction d'inclusion.

Exemple 1.1.9 *Voici un exemple d'application de l'extension naturelle. Les occurrences de* x_1 *sont remplacées par* $[1,2]$ *et les occurrences de* x_2 *par* $[2,6]$*. Les calculs sont ensuite effectués en utilisant les formules de l'arithmétique d'intervalles.*

$$\begin{gathered}\forall x \in \boldsymbol{X} = [1,2] \times [2,6],\ f(x) = x_1 \times x_2^2 - \exp(x_1 + x_2),\\ EN_f(\boldsymbol{X}) = [1,2] \times [2,6]^2 - \exp([1,2] + [2,6]),\\ EN_f(\boldsymbol{X}) = [-2976.9579870417284, 51.91446307681234].\end{gathered}$$

Ainsi, -2976.9579870417284 *est un minorant et* 51.91446307681234 *un majorant de* f *sur* $[1,2] \times [2,6]$*.*

L'analyse d'intervalles peut donc se munir de tous les outils de l'arithmétique classique pour calculer des minorants et des majorants d'une *fonction explicite* sur un intervalle. Malheureusement, les définitions naturelles proposées n'offrent pas les propriétés d'inversibilité et de distributivité. Notons que l'inverse de l'addition dans $\mathbb{I}$ n'est pas la soustraction et l'inverse de la multiplication n'est pas la division. De plus, l'ensemble $\mathbb{I}$ muni de la loi $\times$ est sous-distributive. Ainsi,
$\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{I},\ \mathbf{z} \times (\mathbf{x} + \mathbf{y}) \subseteq \mathbf{z} \times \mathbf{x} + \mathbf{z} \times \mathbf{y}$.

Exemple 1.1.10

$$\begin{aligned} [1,2] + [3,4] &= [4,6], & [4,6] - [3,4] &= [0,3],\\ [1,2] \times [-1,1] + [1,2] \times [3,4] &= [1,10], & [1,2] \times ([-1,1] + [3,4]) &= [2,10],\\ [1,2] \times [3,4] &= [3,8], & [3,8] \div [3,4] &= [3/4, 8/3]. \end{aligned}$$

Définition 1.1.11 *Soit* $f : \boldsymbol{X} \subset \mathbb{R}^n \to \mathbb{R}$. L'image directe *de* f *sur* $\boldsymbol{X}$*, notée* $f(\boldsymbol{X})$*, est l'intervalle de largeur minimale tel que* $\forall x \in \boldsymbol{X}, f(x) \in f(\boldsymbol{X})$*.*

Théorème 1.1.12 *Soit* $f : \boldsymbol{X} \subset \mathbb{R}^n \to \mathbb{R}$*. Si* EN_f *l'extension naturelle de* f *ne possède qu'une seule occurrence de chaque variable, alors l'image directe de* f *sur* $\boldsymbol{X}$ *est exactement* $EN_f(\boldsymbol{X})$*.*

$$\forall\ \mathbf{Z} \subseteq \boldsymbol{X},\ EN_f(\mathbf{Z}) = f(\mathbf{Z}).$$

Donc d'après le théorème précédent, la qualité des minorants et majorants dépend fortement de la forme de l'expression de la fonction. Si une variable apparaît plusieurs fois dans une expression, le minorant calculé sera nécessairement inférieur à la borne inférieure exacte. Ce problème est appelé *problème de dépendance* et est lié à l'arithmétique d'intervalles. C'est pourquoi lorsque l'on étudie des fonctions non convexes avec beaucoup de répétitions de variables, il est préférable de combiner l'extension naturelle avec d'autres techniques de calcul de minorant.

1.1.3 Techniques de calcul de minorant

Comme nous l'avons vu dans la partie précédente, l'extension naturelle aux intervalles est la technique la plus simple pour obtenir un minorant et un majorant d'une fonction explicite sur un intervalle $\mathbf{X} \in \mathbb{I}^n$.

Malheureusement, cette technique a ses limites et se révèle souvent de mauvaise qualité lorsqu'il existe plusieurs occurrences d'une même variable.

Avant tout, pour mesurer la qualité d'une technique de calcul d'encadrement par fonction d'inclusion, on utilise la notion α-convergence :

Définition 1.1.13 *Soit F une fonction d'inclusion de $f : \mathbb{R}^n \to \mathbb{R}^m$. Soit $\boldsymbol{X} \subset \mathbb{R}^n$, $F(\boldsymbol{X})$ représente le résultat de la fonction d'inclusion sur $\boldsymbol{X}$ et $f(\boldsymbol{X})$ l'image directe de f sur $\boldsymbol{X}$. S'il existe une constante K et un $\alpha > 0$ tel que pour tout $\boldsymbol{X}$ avec $w(\boldsymbol{X})$ suffisamment petit tel que*

$$w(F(\boldsymbol{X})) - w(f(\boldsymbol{X})) \leq K w(\boldsymbol{X})^{\alpha}.$$

On dit alors que F est une fonction d'inclusion d'ordre α de f. Si $\alpha = 1$, on parle de fonction d'inclusion du premier ordre ou si $\alpha = 2$ du deuxième ordre.

Définition 1.1.14 *Soit F une fonction d'inclusion de $f : \mathbb{R}^n \to \mathbb{R}^m$. Alors F est une fonction lipschitzienne s'il existe une constante K tel que $\forall \boldsymbol{X} \subset \mathbb{I}^n$*

$$w(F(\boldsymbol{X})) \leq K w(\boldsymbol{X}).$$

Théorème 1.1.15 *[79, 90] L'extension naturelle aux intervalles est une fonction d'inclusion du premier ordre.*

D'autres techniques de fonction d'inclusion consistent à considérer les développements de Taylor au premier ou deuxième ordre. En effet, il est possible d'obtenir des fonctions d'inclusion en considérant le développement de Taylor combiné à l'extension naturelle du gradient de la fonction.

Soit $f : \mathbf{X} \subset \mathbb{R}^n \to \mathbb{R}$ une fonction différentiable et ∇F une fonction d'inclusion du gradient de f, généralement on considère son extension naturelle. Soit x^* un point de $\mathbf{X}$, alors la fonction d'inclusion de Taylor au premier ordre, nous donne :

$$\forall y \in \mathbf{X},\ f(y) \in T_1(x^*, \mathbf{X}) = f(x^*) + (\mathbf{X} - x^*)^T \nabla F(\mathbf{X})$$

Théorème 1.1.16 *[59] Si ∇F est lipschitzienne, alors la forme centrée de Taylor $T_1(mid(\boldsymbol{X}), \boldsymbol{X})$ est une fonction d'inclusion du second ordre.*

La figure 1.1 représente le calcul du minorant obtenu par T_1 d'une fonction à une variable sur un intervalle. En considérant un point de la courbe, toute la courbe se situe nécessairement au-dessus de deux droites de plus forte pente et de plus forte montée. Ainsi, un minorant de la fonction sur l'intervalle s'obtient en prenant

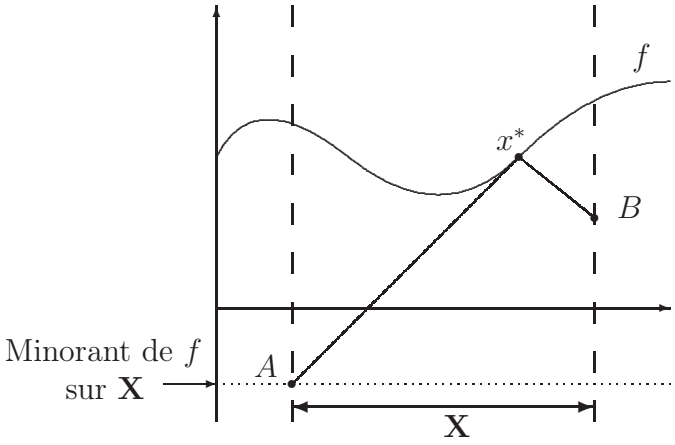


FIGURE 1.1 – Calcul de minorant avec la fonction d'inclusion T_1.

l'intersection la plus basse entre ces droites avec les bornes de l'intervalle (points A et B de la figure 1.1).

Pour améliorer ce minorant, Baumann propose un choix astucieux du point x^* du développement de Taylor pour maximiser le minorant obtenu avec T_1 [15]. Baumann démontre que ce point noté x^B s'obtient lorsque les points A et B de la figure 1.1 sont à la même hauteur. Il peut se calculer facilement grâce à la formule générale suivante :
Soit $x^B = (x_1^B, x_2^B, \ldots, x_n^B)$ le centre de Baumann de f sur $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ avec $x_i^B = \dfrac{\underline{\mathbf{x}_i}\overline{\nabla F(\mathbf{X})_i} - \overline{\mathbf{x}_i}\,\underline{\nabla F(\mathbf{X})_i}}{\overline{\nabla F(\mathbf{X})_i} - \underline{\nabla F(\mathbf{X})_i}}$ pour $i = 1, 2, \ldots, n$. La valeur du minorant f_{zB} de f sur $\mathbf{X}$ est alors :

$$f_{zB} = f(x^B) + \sum_{i=1}^{n} \frac{(\overline{\mathbf{x}_i} - \underline{\mathbf{x}_i})\underline{\nabla F(\mathbf{X})_i}\,\overline{\nabla F(\mathbf{X})_i}}{\overline{\nabla F(\mathbf{X})_i} - \underline{\nabla F(\mathbf{X})_i}}.$$

Cette technique de calcul de fonctions d'inclusion par développement de Taylor peut facilement s'étendre à l'ordre n pour les fonctions $f : \mathbb{R} \to \mathbb{R}$. Voici par exemple le cas général pour une fonction f à une seule variable. Notons $f^{(n)}$ la dérivé $n^{\text{ième}}$ de f, $F^{(n)}$ sa fonction d'inclusion et x^* un point de $\mathbf{X}$, alors pour tout n :

$$\forall x \in \mathbf{X}, f(x) \in T_n(x^*, \mathbf{X}) = f(x^*) + (x - x^*)f'(x^*) + \cdots + \frac{(x - x^*)^n}{n!}F^{(n)}(\mathbf{X}).$$

Théorème 1.1.17 *[91, p42] Si f est deux fois différentiable, et si f'' a une fonction d'inclusion bornée F'', alors la forme de Taylor T_2 est une fonction d'inclusion du deuxième ordre.*

Il existe de nombreuses autres techniques de calcul de minorant pouvant être combinées : la technique du cerf-volant [106], la technique du simplexe admissible

[72], etc... L'article [43] effectue une étude comparative des techniques de Taylor et de la technique du simplexe admissible. L'étude montre que la technique du simplexe admissible fournit de meilleurs minorants.

1.2 Algorithme de Branch and Bound par intervalles : IBBA

De nos jours, les algorithmes de Branch and Bound ont pris une place non négligeable dans le monde de l'optimisation globale. Les raisons de cet engouement sont simples, c'est un principe à la fois basique et général sur lequel peuvent venir se greffer de nombreuses idées pour améliorer la convergence. Dans la suite de notre étude, nous ne nous intéresserons qu'au Branch and Bound basé sur l'analyse d'intervalles, autrement appelé les algorithmes de Branch and Bound par intervalles.

Les premiers algorithmes de Branch and Bound par intervalles sont nés dans les années 70. Les premières études étaient surtout très théoriques et assez avares en résultats numériques. Ceci devait être certainement dû à la capacité des ordinateurs qui à l'époque ne permettait de disposer que de quelques méga-octets de mémoire.

Le premier algorithme de Branch and Bound par intervalles fut celui de Moore-Skelboe en 1974 [100], puis Ichida et Fujii y rajoutèrent le test au point milieu en 1979 [47], et Hansen [42], Kearfott [51], Ratz [40] et beaucoup d'autres continuèrent à développer l'algorithme en y incluant des tests de monotonies, de concavité, etc., tout en considérant des problèmes de plus en plus généraux. Dans [52, Table 5.1], Kearfott retrace cette évolution.

L'algorithme que nous allons utiliser est issu du livre de Ratschek et Rokne [91] qui compile la plupart des techniques de base (ou qui sont devenues basiques). Cet algorithme permet de résoudre des problèmes mixtes d'optimisation globale avec contraintes. C'est-à-dire qu'il recherche l'optimum global d'une fonction sous contraintes d'égalité et d'inégalité avec des variables continues ou entières. Ces problèmes peuvent s'écrire sous la forme (PB). Toutes les fonctions f, g_i et h_j peuvent être non-convexes. La seule condition est que ces fonctions doivent posséder une *expression explicite*, pour pouvoir utiliser l'arithmétique d'intervalles. Le domaine d'étude $\mathbf{X}$ est un sous-ensemble convexe de $\mathbb{R}^n$.

$$\left\{\begin{array}{lll} \min\limits_{x\in\mathbf{X}\subset\mathbb{R}^n} & f(x) & \\ \text{s.t.} & g_i(x)\leq 0\ , & \forall i\in\{1,\ldots,p\},\\ & h_j(x)=0\ , & \forall j\in\{1,\ldots,q\}. \end{array}\right. \tag{PB}$$

L'algorithme 1 schématise la structure générale des Branch and Bound par intervalles.

Pour résumer, il y a quatre étapes que nous réalisons dans cet ordre : l'extraction, la coupe, l'analyse, l'insertion. Certains algorithmes effectuent ces étapes dans un autre ordre : l'extraction, l'analyse, la coupe, l'insertion. L'ordre n'a pas d'impact sur

Algorithme 1 Structure de l'algorithme de Branch and Bound par intervalles

1: Initialisation de la structure de données $\mathcal{L}$.
2: **répéter**
3: Extraire un élément V de $\mathcal{L}$.
4: Division de V en plusieurs sous-domaines V_i, avec $i = 1, 2, \ldots$
5: **pour** chaque sous-domaine V_i **faire**
6: Utilisation de techniques d'accélération,
7: Vérification des contraintes,
8: Recherche d'une bonne solution réalisable,
9: Élimination ou Insertion de V_i dans $\mathcal{L}$,
10: **fin pour**
11: **jusqu'à** un critère d'arrêt.

le comportement de l'algorithme. Néanmoins, l'ordre que nous avons choisi permet de réduire les insertions et les extractions. Car, dans notre version, la structure de données contenant tous les éléments stockés, que nous appelons $\mathcal{L}$, ne contient que des éléments analysés, alors que la structure $\mathcal{L}$ de l'autre version peut contenir des éléments devant être supprimés directement après analyse. De plus, notre version laisse la possibilité de paralléliser facilement la boucle de la ligne 5 à 10 sur un bi-processeur.

L'extraction d'un élément de $\mathcal{L}$ peut se faire suivant plusieurs critères. Pour gagner en performance, il est très important d'accorder ce critère avec la structure de données. Si un parcours en profondeur d'abord est choisi, il est préférable que $\mathcal{L}$ soit une liste LIFO : insertion en tête de liste et extraction en tête. Si un parcours en largeur est choisi, $\mathcal{L}$ devrait être une liste FIFO : insertion en queue de liste et extraction en tête. Si les éléments sont choisis suivant une fonction coût, la meilleure structure pour stocker les éléments est un *minimier*, cf. remarque 1.2.1.

Concernant l'étape 4 de l'algorithme 1, il existe plusieurs techniques pour diviser l'élément V en deux sous-boites [27]. La technique la plus couramment utilisée consiste à sélectionner la composante ayant la plus grande largeur et à la diviser en deux en son milieu pour générer deux sous-domaines.

La boucle FOR de la ligne 5 à 10 constitue le coeur de l'algorithme 1.

ligne 6 : Les techniques d'accélération pouvant être ajoutées sont nombreuses. Nous en détaillerons quelques-unes dans la section 1.3.

ligne 7 : La vérification des contraintes se fait généralement par analyse d'intervalles. Pour chaque contrainte d'inégalité $g(x) \leq 0$, on calcule une fonction d'inclusion de g qui nous donne un intervalle $G(V_i)$ contenant toutes les valeurs de la fonction g sur le domaine V_i : (i) si $G(V_i) \leq 0$, on est sûr que l'élément V_i et tous ses sous-domaines vérifient la contrainte ; (ii) si $G(V_i) > 0$, on est sûr que chaque élément de V_i ne vérifie pas la contrainte ; (iii) sinon, on ne peut rien conclure. Pour les contraintes d'égalité, il est nécessaire d'inclure une précision sur la contrainte. En effet, en analyse d'intervalles, en évaluant une fonction d'inclusion en un point flottant, il est impossible d'obtenir un

intervalle de largeur nulle, car toutes les erreurs d'arrondi intrinsèques au format flottant IEEE-754 [2] sont prises en compte. Il est donc impossible de vérifier exactement une contrainte d'égalité, et c'est aussi le cas en calcul flottant, par exemple avec la fonction *fmincon* de *Matlab*. Pour cela, toutes les contraintes d'égalité de type $h(x) = 0$ sont remplacées par une contrainte de type $h(x) \in [-\epsilon, \epsilon]$, avec ϵ choisi par l'utilisateur. Considérons donc $H(V_i)$ le résultat d'une fonction d'inclusion de h sur V_i : (i) si $H(V_i) \subset [-\epsilon, \epsilon]$, la contrainte est vérifiée pour tout x inclus dans V_i ; (ii) si $H(V_i) \cap [-\epsilon, \epsilon] = \emptyset$, la contrainte est insatisfaite pour tout x inclus dans V_i ; (iii) sinon, on ne peut rien conclure.

ligne 8 : La recherche d'une bonne solution réalisable consiste, dans le cas simple, à évaluer la fonction objectif au point milieu de V_i. Si ce point vérifie les contraintes, on le compare à la meilleure solution courante du problème général et on la met à jour si celle-ci est meilleure. Pour compléter cette évaluation, il est possible d'effectuer quelques pas de descente locale dans V_i pour trouver un meilleur point que le milieu.

ligne 9 : Pour en finir, on élimine l'élément V_i si une contrainte est sûre d'être insatisfiable ou si après évaluation d'un minorant de la fonction objectif sur V_i, celui-ci est supérieur à la meilleure solution courante, c'est-à-dire que l'on est sûr que le minimum global n'appartient pas à V_i.

Concernant les conditions d'arrêt de l'algorithme, il existe deux tests d'arrêt prouvant la terminaison et la convergence de l'algorithme : (i) sur la valeur de la fonction objectif, lorsque l'écart entre la valeur du minimum courant et le plus petit des minorants des éléments stockés dans $\mathcal{L}$ est inférieur à une précision relative ϵ_f souhaitée par l'utilisateur ; (ii) sur la largeur des éléments encore stockés dans $\mathcal{L}$, lorsque la largeur de tous les éléments stockés dans $\mathcal{L}$ est inférieure à une précision $\epsilon_{\mathcal{L}}$ souhaitée par l'utilisateur.

Remarque 1.2.1 *Une attention particulière doit être apportée à la structure de données $\mathcal{L}$. Jusqu'à présent, il était commun d'utiliser une simple* liste triée. *La complexité de l'insertion d'un élément dans le pire des cas est linéaire en fonction du nombre d'élément n, l'extraction est directe, et le parcours de la liste pour éliminer des éléments est linéaire en n. Cependant en réalisant des tests numériques, il s'est avéré qu'une structure de* minimier *était beaucoup plus efficace. Un* minimier *est une structure de tas binaire et parfaitement équilibré. Tous ses niveaux sont pleins excepté le dernier. Les nœuds sont ordonnés suivant un ordre partiel, c'est-à-dire qu'un nœud est plus grand que ses deux fils et plus petit que son père. Cette structure de* minimier *minimise la hauteur du tas généré, ce qui est très important pour notre algorithme, car la taille de $\mathcal{L}$ peut grossir de façon très importante et déséquilibrer les structures par arbres non équilibrés. Ce gain en performance peut se remarquer en analysant l'ordre de grandeur de la complexité des opérations usuelles. Soit n le nombre d'éléments stockés dans $\mathcal{L}$: (i) l'insertion a une complexité de* $\log_2(n)$ *; (ii) l'extraction* $\log_2(n)$ *(iii) l'élimination des éléments* $n \log_2(n)$ *(voir ligne 14 de l'algorithme 2).*

Algorithme 2 Interval Branch and Bound Algorithm : IBBA

1: Soit $\mathbf{X}$ = le domaine initial dans lequel le minimum global est recherché,
$\tilde{f} = +\infty$, désigne le majorant courant du minimum global,
$R = [0, \ldots, 0]$, le vecteur de vérification des contraintes,
$\mathcal{L} = \{(\mathbf{X}, -\infty, R)\}$, l'initialisation de la structure de données pour stocker les éléments,
2: **répéter**
3: Extraire de $\mathcal{L}$ un élément $V = (\mathbf{Z}, f_z, R)$,
4: Division : $V_1 = (\mathbf{Z}_1, f_{z1}, R_1)$, $V_2 = (\mathbf{Z}_2, f_{z2}, R_2)$,
avec $R_1 = R_2 = R$, $f_{z1} = f_{z2} = f_z$, $\mathbf{Z} = \mathbf{Z}_1 \cap \mathbf{Z}_2$ et $\mathbf{Z}_1 \cup \mathbf{Z}_2 = \emptyset$,
5: **pour** $j = 1$ à 2 **faire**
6: Technique de réduction de $\mathbf{Z}_j$ par propagation des contraintes (voir section 1.3.2),
7: Mise à jour de f_{zj} : un minorant de la fonction objectif sur $\mathbf{Z}_j$,
8: Vérification des contraintes et mise à jour de R_j,
9: **si** $\left(f_{zj} \leq \tilde{f} - \epsilon_f \max(|\tilde{f}|, 1)\right)$ et aucune contrainte est insatisfaite **alors**
10: Insertion $(\mathbf{Z}_j, f_{zj}, R_j)$ dans $\mathcal{L}$,
11: $\tilde{f} = \min\{\tilde{f}, f(mid(\mathbf{Z}_j))\}$, si et seulement si m satisfait toutes les contraintes,
12: **si** $\tilde{f}$ a changé **alors**
13: $\tilde{z} = mid(\mathbf{Z}_j)$
14: Retirer de $\mathcal{L}$ tous les éléments $(\mathbf{Z}, f_z, R)$ tel que $\tilde{f} - \epsilon_f \max(|\tilde{f}|, 1) < f_z$
15: **finsi**
16: **finsi**
17: **fin pour**
18: **jusqu'à** $\left(\tilde{f} - \min_{(\mathbf{Z}, f_z, R) \in \mathcal{L}} f_z \leq \epsilon_f \max(|\tilde{f}|, 1)\right)$ ou lorsque $\mathcal{L} = \emptyset$

L'algorithme que nous avons utilisé et développé s'appelle *IBBA*, pour Interval Branch and Bound Algorithm. Il est donc basé sur l'algorithme 1 et résumé dans l'algorithme 2. Nous avons voulu rester très simples dans le principe général de l'algorithme en incluant peu de techniques d'accélération, tout en le rendant très modulaire. Cette modularité nous a été très utile par la suite, car cela a facilité l'étude des techniques d'accélération en isolant, pour chacune d'entre elles, leur impact sur la convergence, sur les performances et les interactions qui ont pu améliorer la résolution des problèmes. L'algorithme 2 permet de résoudre des problèmes d'optimisation globale de type (PB). Si une solution réalisable existe, l'algorithme retourne dans $\tilde{f}$ un majorant de la valeur du minimum global et dans $\tilde{z}$ une solution telle que $f(\tilde{z}) = \tilde{f}$. Voici quelques détails supplémentaires sur l'algorithme 2.

ligne 1 : La structure de données $\mathcal{L}$ est un minimier, cette structure est détaillée dans la remarque 1.2.1. Chaque élément V stocké dans $\mathcal{L}$ est composé de trois objets. Notons $V = (\mathbf{Z}, f_z, R)$ où $\mathbf{Z}$ est un sous-domaine convexe de $\mathbf{X}$, f_z est le plus grand minorant connu de la fonction objectif sur $\mathbf{Z}$ et R est un vecteur de

booléen indiquant pour chaque contrainte du problème si elle est vérifiée pour tous les éléments de $\mathbf{Z}$. En d'autres termes pour chaque contrainte d'inégalité g_i, si $R(i) = 1$, alors $\forall x \in \mathbf{Z}, g_i(x) \leq 0$, et si $R(i) = 0$, on ne peut rien conclure. Le principe est le même pour les contraintes d'égalité. Ce vecteur R est très intéressant, car si une contrainte est vraie pour un domaine, alors celle-ci reste vraie pour tous ses sous-domaines. Les éléments sont triés dans $\mathcal{L}$ par rapport à leur valeur de f_z. Ainsi, l'élément ayant le plus petit f_z se situe à la racine pour faciliter l'extraction, ligne 3.

ligne 4 : Pour l'étape de division, le domaine sélectionné $\mathbf{Z}$ est divisé en deux sous-domaines $\mathbf{Z}_1$ et $\mathbf{Z}_2$ en coupant la composante de largeur maximale en son milieu. Le meilleur minorant connu f_z et le vecteur de vérification des contraintes R sont recopiés dans les fils V_1 et V_2, car ils restent toujours valables pour les sous-domaines $\mathbf{Z}_1$ et $\mathbf{Z}_2$.

ligne 6 : Pour éviter que les différentes techniques d'accélération n'interfèrent dans nos études, nous n'avons inclus dans l'algorithme de base que la technique de propagation des contraintes de la section 1.3.2.

ligne 7 : L'évaluation d'un bon minorant de la fonction objectif f se fait par analyse d'intervalles en calculant la fonction d'inclusion de f sur $\mathbf{Z}_i$.

ligne 8 : Pour chaque contrainte dont la variable R_i associée est nulle, on vérifie si la contrainte est toujours vraie ou toujours fausse. Des minorants et majorants de chaque contrainte sont calculés par analyse d'intervalles et la vérification s'effectue comme à la ligne 7 de l'algorithme 1.

ligne 11 : Pour la mise à jour et la recherche de la meilleure solution, nous évaluons simplement la fonction au milieu du domaine considéré.

ligne 14 : La phase d'élimination consiste à retirer de la structure de données $\mathcal{L}$ tous les éléments qui sont sûrs de ne pas contenir une solution meilleure que la solution courante moins la précision demandée sur la fonction objectif.

ligne 18 : Lorsque l'algorithme termine, si $\tilde{f} = +\infty$ et $\tilde{z}$ est non initialisé, alors cela veut dire que le problème de départ n'a pas de solution réalisable sur $\mathbf{X}$ avec la précision souhaitée sur les contraintes. Si $\tilde{f} < +\infty$, alors la valeur du minimum global du problème (PB) est comprise dans $[\tilde{f} - \epsilon_f \max(|\tilde{f}|, 1), \tilde{f}]$ et $\tilde{z}$ est une solution réalisable telle que $f(\tilde{z}) = \tilde{f}$. On dit que ϵ_f est la précision relative demandée sur la fonction objectif f et $\tilde{f}$ est le *minimum ϵ_f-global* du problème (PB).

Cette version de l'algorithme de Branch and Bound par intervalles a été implémentée en Fortran 90. Ce nouveau code est basé sur celui qu'a développé Messine depuis le début de sa thèse en 1997 [68] et qu'il n'a jamais cessé d'améliorer depuis [20, 21, 43, 60, 69, 72, 75, 85]. Ce code a surtout été amélioré et utilisé pour résoudre des problèmes de conception d'actionneurs électriques [32, 35, 73, 74, 89]. D'autres améliorations techniques du code ont été apportées durant notre étude, notamment avec l'utilisation d'un minimier, l'utilisation de l'arithmétique d'intervalles incluse dans le compilateur SUN f90 et une réorganisation plus modulaire pour inclure plus facilement les techniques d'accélération que nous étudierons par la suite.

Ce code *IBBA* nous servira dans la suite de ce travail pour réaliser tous les tests numériques.

1.3 Techniques d'accélération

Il existe de nombreuses techniques pour accélérer la convergence des algorithmes de Branch and Bound par intervalles. Dans cette section, on trouvera un aperçu des techniques les plus couramment utilisées et qui ont déjà prouvé leur efficacité.

1.3.1 Test de monotonie

Le test de monotonie consiste à vérifier si la fonction objectif est monotone sur un sous-domaine **Z**. Cette technique s'utilise généralement pour résoudre des problèmes sans contrainte ou sur des sous-domaines pour lesquels toutes les contraintes sont vérifiées.

Pour effectuer ce test, on calcule une fonction d'inclusion du gradient de la fonction étudiée. Si la fonction d'inclusion prouve que le gradient est toujours positif ou toujours négatif sur le sous-domaine **Z** alors la fonction est monotone sur **Z** et il est donc facile d'en déduire son minimum sur **Z**.

1.3.2 Techniques de réduction de domaine par propagation des contraintes

Les méthodes de réduction permettent de réduire l'intervalle d'étude en utilisant les contraintes pour ainsi converger plus rapidement. Cette technique permet de faire le lien entre la programmation par contraintes et l'optimisation globale. Toutes les avancées en programmation par contraintes peuvent être incluses dans un algorithme de Branch and Bound par intervalles pour repérer des solutions réalisables ou pour réduire le domaine de recherche.

En optimisation globale, la première technique de propagation des contraintes a été développée par Hansen [42]. Elle consiste à utiliser les fonctions d'inclusion T_1 basées sur le développement de Taylor à l'ordre 1 de chaque contrainte, pour obtenir un système linéaire d'intervalles, c'est-à-dire un système linéaire dans lequel toutes les constantes sont des intervalles. Ensuite, en utilisant un algorithme de Gauss-Seidel adapté à l'arithmétique d'intervalles, on essaye de réduire le domaine d'étude, afin de déterminer les plus petits intervalles contenant toutes les solutions réalisables. Cette technique a déjà été longuement étudiée et à déjà fait ses preuves quant à son efficacité. Elle permet, en effet, de considérer toutes les contraintes à la fois. Néanmoins, l'expression du gradient est nécessaire pour toutes les contraintes et la matrice d'intervalles du système linéaire a besoin d'être préconditionnée pour que l'algorithme de Gauss-Seidel soit efficace.

La technique que nous avons préféré utiliser est beaucoup plus simple dans sa programmation et dans son principe. Nous utilisons une version développée par

Messine [68, 70], et est applicable à tout *problème explicite*. Elle consiste à faire propager les contraintes en se basant sur les *arbres de calcul*, c'est-à-dire l'arbre correspondant à l'expression analytique de la contrainte.

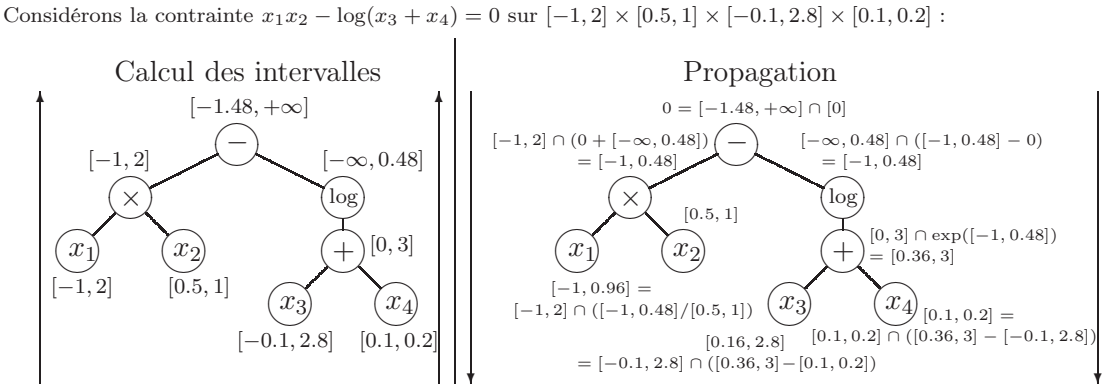


FIGURE 1.2 – Technique de propagation des contraintes.

Pour chaque contrainte, on construit l'*arbre de calcul*. Sur chaque nœud de l'arbre, on calcule l'intervalle correspondant au calcul intermédiaire, comme décrit sur la partie gauche de la figure 1.2. Puis, on intersecte le résultat de la racine avec l'intervalle imposé par la contrainte et on propage ce résultat en effectuant toutes les opérations inverses pour descendre dans l'arbre comme le décrit la partie droite de la figure 1.2. Ainsi, étant donné que les feuilles correspondent soit à des constantes soit à des variables, il est possible de réduire l'intervalle de chaque variable qui apparaît dans une contrainte. Le procédé est réitéré au maximum 100 fois ou jusqu'à ce que plus aucune réduction ne soit effectuée sur les variables.

De plus, si l'on obtient un intervalle vide sur l'un des nœuds, cela signifie que la contrainte n'est pas satisfaite pour cet élément $\mathbf{X}$ et qu'il peut donc être éliminé.

Cette méthode est décrite dans l'algorithme 3 et s'intègre facilement dans l'algorithme 2 en ligne 6.

Remarque 1.3.1 *Une autre version de ce type d'algorithme a été intégrée dans le logiciel* Numerica *sous le nom* HC4 *[18, 38, 104] ainsi que dans le solveur* Couenne *sous le nom de* FBBT *[16, 17]. D'autres techniques de propagation des contraintes existent. Certaines sont basées à la fois sur des structures d'arbres plus complexes, par exemple les graphes directs acycliques (DAG) et des combinaisons de plusieurs méthodes pour calculer des bornes [63, 107].*

Ainsi, cette technique ne nécessite aucun calcul de gradient et travaille directement sur la forme complète des contraintes. Néanmoins, étant donné que les contraintes sont considérées une par une, il est parfois conseillé d'effectuer plusieurs fois de suite l'algorithme 3, pour que les réductions apportées par une contrainte puissent être amplifiées par les autres.

Algorithme 3 Algorithme de propagation des contraintes

Initialisation avant la première utilisation de l'algorithme :
Construction de l'arbre de calcul de chaque contrainte. Les arbres ne sont construits qu'une seule fois puis réutilisés pour économiser un nombre important d'allocations mémoires.
Soit $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ un sous-domaine.
$W = w(\mathbf{X}) + 1$; $k = 1$.
tantque $w(\mathbf{X}) < W$ et $k < 100$ **faire**
  $W = w(\mathbf{X})$; $k = k + 1$.
  pour tout i **faire**
    Considérons la contrainte "$C_i(x) = \mathbf{b_i}$" avec C_i l'arbre de la contrainte i et $\mathbf{b_i}$ un intervalle.
    Initialisation des intervalles de chaque feuille avec les $\mathbf{x}_i$.
    Mise à jour des valeurs des nœuds de tout l'arbre en effectuant les calculs par arithmétique d'intervalles.
    On intersecte l'intervalle obtenu à la racine de l'arbre avec $\mathbf{b}$.
    pour chaque nœud P de l'arbre en partant de la racine **faire**
      si P contient un opérateur **alors**
        On propage la réduction sur ses fils en effectuant les opérations inverses :
        par exemple si $P = F_1 \times F_2$ alors
        $F_1 = (P/F_2) \cap F_1$ et $F_2 = (P/F_1) \cap F_2$.
      sinon { P est une feuille}
        On réduit l'intervalle de la variable associée : $\mathbf{x}_i = P \cap \mathbf{x}_i$.
      finsi
      si L'intervalle du nœud P est vide, **alors STOP**
      Aucun élément du sous-domaine $\mathbf{X}$ ne satisfait la contrainte considérée.
    fin pour
  fin pour
fin tantque

1.3.3 Reformulation-Linearization Technique (RLT)

Les techniques de reformulation ont été utilisées en optimisation globale depuis plusieurs années [4, 8, 46, 54, 61, 65, 88, 96, 102]. L'idée principale de toutes ces techniques de reformulation consiste à approximer un problème général par une relaxation linéaire. Ainsi, la résolution du problème linéaire fournit un minorant ou un certificat d'infaisabilité du problème d'origine.

La méthode de relaxation la plus utilisée est la Technique de Reformulation-Linéarisation (RLT) [96, 99]. Cette technique est applicable principalement aux *problèmes explicites* [97]. Dans la version classique de RLT, *l'arbre de calcul* de chaque fonction est divisé et chaque nœud est remplacé par une nouvelle variable. Ainsi, pour chaque fonction, chaque terme non-affine ajoute une nouvelle variable et au

minimum une contrainte à la relaxation linéaire.

Par exemple, pour linéariser $x \times y$, la technique la plus couramment utilisée est celle des inégalités de McCormick [67]. Elle permet de construire l'enveloppe convexe du produit de deux variables sur un intervalle, grâce à 4 inégalités linéaires. Cela permet donc de linéariser tous les produits binaires [77].

Kearfott et Hongthong dans [53, 54] ont développé une technique de calcul de borne basée sur une relaxation linéaire. Le principe consiste à reformuler le problème de départ en ajoutant de nouvelles variables pour chaque terme non linéaire, puis à approximer chaque nouveau terme par plusieurs équations linéaires. Cette technique fut ensuite incluse dans un algorithme de Branch and Bound par intervalles nommé *GlobSol*. Ainsi à chaque itération de la boucle principale de l'algorithme, le problème général est reformulé selon cette technique, puis après résolution du système linéaire, on peut en déduire un bon minorant de la fonction objectif.

Néanmoins, les résultats numériques ne furent pas très concluants [53]. En effet, la résolution du problème linéaire peut prendre trop de temps, car le système linéaire généré peut atteindre une taille assez importante. Ainsi, malgré l'amélioration des minorants et la diminution du nombre d'itérations, la résolution globale des problèmes peut être plus rapide sans la technique.

Chapitre 2

L'Arithmétique Affine Standard et ses Extensions

Dans ce chapitre, nous étudions différentes façons de développer des arithmétiques basées sur le même principe que l'arithmétique affine standard introduite par Comba et Stolfi [23]. Les extensions permettent de réduire la taille des formes affines. Les combinaisons entre les arithmétiques améliorent la qualité des encadrements et différentes façons de les rendre robustes y sont développées.

2.1 L'arithmétique affine standard

L'arithmétique affine est une autre approche pour calculer des minorants et des majorants de fonction explicite sur un intervalle. Elle fut introduite en 1993 par Comba et Stolfi [23]. L'idée est de construire une fonction d'inclusion basée sur des formes affines. Ceci a pour but de limiter les erreurs liées notamment aux répétitions des variables que les fonctions d'inclusion basées sur l'arithmétique d'intervalles gèrent mal. Ces méthodes étaient jusqu'alors utilisées en infographie pour détecter plus précisément la zone d'intersection de deux surfaces [29, 30, 66, 101].

Le principe consiste donc à garder de l'information linéaire pendant le calcul. C'est-à-dire que toutes les quantités vont s'écrire comme l'addition d'une constante et d'une combinaison linéaire d'intervalles $[-1, 1]$ (cf équation (2.1)).

Une forme affine standard peut s'écrire comme suit, où x est une quantité, les coefficients x_i sont des nombres flottants et ϵ_i sont des variables réelles dont les valeurs ne sont pas connues mais appartiennent à $[-1, 1]$. Nous noterons pour tout $i \in \{0, \ldots, n\}$, $x_i \in \mathbb{R}$ et $\epsilon_i = [-1, 1]$ [101] :

$$\begin{gathered} \widehat{x} = x_0 + \sum_{i=1}^{n} x_i \epsilon_i, \\ \text{avec } \forall i \in \{0, \ldots, n\}, x_i \in \mathbb{R} \text{ et } \forall i \in \{1, \ldots, n\}, \epsilon_i = [-1, 1]. \end{gathered} \tag{2.1}$$

Bien entendu, l'arithmétique affine et l'arithmétique d'intervalles restent très proches et la conversion pour passer de l'une à l'autre forme se fait comme suit :

Intervalle $\longrightarrow$ Forme Affine **Forme Affine $\longrightarrow$ Intervalle**

$$\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}] \longrightarrow \widehat{x} = \frac{\underline{\mathbf{x}} + \overline{\mathbf{x}}}{2} + \frac{\underline{\mathbf{x}} - \overline{\mathbf{x}}}{2}\epsilon_k \quad (2.2)$$

avec ϵ_k une nouvelle variable.

$$\widehat{x} = x_0 + \sum_{i=1}^{n} x_i\epsilon_i \longrightarrow \mathbf{x} = x_0 + \left(\sum_{i=1}^{n} \mid x_i \mid\right) \times [-1, 1]. \quad (2.3)$$

Tout comme l'arithmétique d'intervalles, les opérations élémentaires sont aussi redéfinies pour pouvoir considérer des formes affines. Les opérateurs affines sont facilement redéfinis puisqu'aucune approximation n'est nécessaire.

$$\widehat{x} \pm \widehat{y} = (x_0 \pm y_0) + \sum_{i=1}^{n}(x_i \pm y_i)\epsilon_i,$$
$$a \pm \widehat{x} = (a \pm x_0) + \sum_{i=1}^{n} x_i\epsilon_i,$$
$$a \times \widehat{x} = ax_0 + \sum_{i=1}^{n} ax_i\epsilon_i.$$

En revanche, la multiplication, l'inverse, l'exponentielle, le logarithme et la racine carrée n'étant pas des opérations affines, une erreur est générée afin de linéariser la forme résultante.

En arithmétique affine standard, l'idée est de rajouter une variable contenant une majoration de l'erreur générée par les termes non linéaires. Pour la multiplication, seul le produit des deux sommes est non linéaire. Il est majoré grâce à la relation (2.4), avec ϵ_{n+1} une nouvelle variable $[-1, 1]$.

$$\sum_{i=1}^{n} x_i\epsilon_i \times \sum_{i=1}^{n} y_i\epsilon_i \leq \left(\sum_{i=1}^{n} \mid x_i \mid \times \sum_{i=1}^{n} \mid y_i \mid\right) \epsilon_{n+1}, \quad (2.4)$$

$$\text{avec } \forall i \in \{1, \ldots, n+1\}, x_i \in \mathbb{R} \text{ et } \epsilon_i = [-1, 1].$$

Ainsi, la multiplication de deux formes affines devient, avec ϵ_{n+1} une nouvelle variable :

$$\widehat{x} \times \widehat{y} = x_0y_0 + \sum_{i=1}^{n}(x_0y_i + x_iy_0)\epsilon_i + \left(\sum_{i=1}^{n} \mid x_i \mid \times \sum_{i=1}^{n} \mid y_i \mid\right) \epsilon_{n+1}.$$

Pour le calcul des fonctions unaires, De Figueiredo et Stolfi [101] proposent deux moyens de trouver une approximation linéaire : l'approximation de Chebyshev et l'approximation min-range, voir figure 2.1. L'approximation de Chebyshev est la linéarisation qui minimise l'erreur maximale. L'approximation min-range minimise l'image de l'approximation linéaire. Les reformulations affines peuvent être notées comme suit :

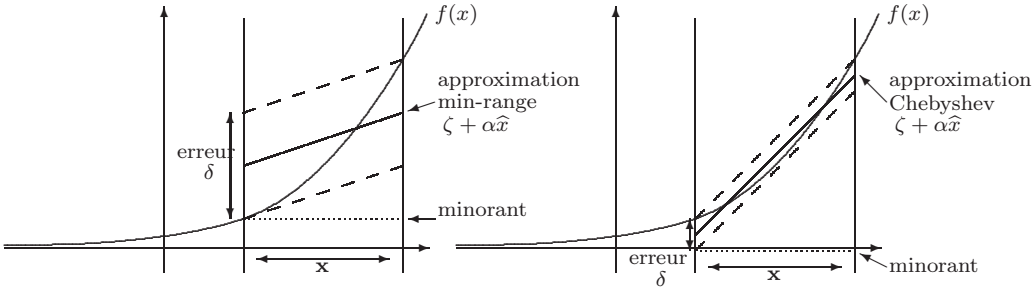


FIGURE 2.1 – Approximation affine par la méthode de Chebyshev et min-range.

$$\widehat{f}(\widehat{x}) = \zeta + \alpha\widehat{x} + \delta\epsilon_{n+1}, \quad \text{avec } \widehat{x} \text{ une forme affine donnée et } \zeta \in \mathbb{R}, \alpha \in \mathbb{R}^+, \delta \in \mathbb{R}^+. \tag{2.5}$$

Ainsi sur la partie droite de la figure 2.1, la linéarisation de Chebyshev donne l'approximation affine avec l'erreur δ minimale, mais le minorant calculé est très inférieur à la véritable borne inférieure. Sur la partie gauche de la figure 2.1, la linéarisation min-range est beaucoup moins efficace pour estimer les dépendances linéaires entre les variables, mais le minorant calculé est meilleur. Dans notre implémentation tout comme dans celle de De Figueiredo et Stolfi, nous avons décidé de privilégier la linéarisation min-range, car il s'avère que pour le calcul de minorant, celle-ci donne de bien meilleurs résultats.

En effet, en utilisant les conversions des équations (2.2) et (2.3), il est possible de construire une fonction d'inclusion ; tous les intervalles de chaque variable sont convertis en forme affine, les calculs sont effectués en utilisant l'arithmétique affine et le résultat est ensuite reconverti en intervalles ; ceci permet d'obtenir un minorant et un majorant de la fonction sur l'intervalle de départ [69, 101]. La fonction d'inclusion générée ne peut être comparée de manière théorique avec l'extension naturelle de l'arithmétique d'intervalles. Néanmoins, les études faites par De Figueiredo et Stolfi [101], Messine [69] et Messine et Touhami [103] ont montré qu'en pratique, appliquée à l'optimisation globale, l'arithmétique affine est très généralement plus efficace pour calculer des minorants que l'utilisation directe de l'arithmétique d'intervalles.

Exemple 2.1.1 *Calcul de minorant par l'arithmétique affine :*

$$\forall x \in [1,2] \times [3,4] \ , \ f(x) = x_1^2 + x_2^2 - x_1 x_2,$$

$$x_1 \to \frac{3}{2} + \frac{1}{2}\epsilon_1, \qquad x_2 \to \frac{7}{2} + \frac{1}{2}\epsilon_2,$$

$$f(\widehat{x}) = \frac{37}{4} - \frac{1}{4}\epsilon_1 + \frac{11}{4}\epsilon_2 + \frac{1}{4}\epsilon_3 + \frac{1}{4}\epsilon_4 - \frac{1}{4}\epsilon_5.$$

En Arithmétique d'Intervalles, $\forall x \in [1,2] \times [3,4] \ , \ f(x) \in [2, 17]$.
En Arithmétique Affine, $\forall x \in [1,2] \times [3,4] \ , \ f(x) \in [\mathbf{5.5}, \mathbf{13}]$.
En considérant l'image directe, $\forall x \in [1,2] \times [3,4] \ , \ f(x) \in [7, 13]$.

Néanmoins, l'implémentation de l'arithmétique affine standard est difficile, car le nombre des variables augmente avec le nombre d'opérations non-affines. Il est donc impossible de prévoir la taille des formes affines manipulées. Or notre objectif est d'utiliser cette arithmétique dans notre algorithme de Branch and Bound par intervalles, nous n'avons donc aucun contrôle sur le nombre d'opérations non-affines. C'est pourquoi nous avons préféré utiliser les extensions de l'arithmétique affine.

2.2 Extensions de l'arithmétique affine : AF1, AF2

L'arithmétique affine standard telle qu'elle a été décrite par De Figueiredo et Stolfi dans [101] introduit une nouvelle variable à chaque fois qu'une opération non-affine est effectuée. Ainsi, la taille de la forme affine n'est pas fixe et dépend de la forme de l'expression de l'équation. Dans [69], Messine propose deux extensions de l'arithmétique affine standard, notées AF1 et AF2. Ces formes affines étendues permettent de fixer le nombre de variables et de garder l'erreur générée par l'approximation des termes non-affines :

- La première forme affine AF1 est basée sur le même principe que l'arithmétique affine standard, à l'exception que cette fois-ci tous les termes générés par les erreurs d'approximation sont cumulés dans un seul terme. Ainsi, le nombre de variables ne varie pas. Une forme AF1 s'écrit de la manière suivante :

$$\begin{gathered}\widehat{x} = x_0 + \sum_{i=1}^{n} x_i\epsilon_i + x_\pm\epsilon_\pm, \\ \text{avec } \forall i \in \{0,1,\ldots,n\}, x_i \in \mathbb{R},\ \epsilon_i = [-1,1]\ ,\ x_\pm \in \mathbb{R}^+ \text{ et } \epsilon_\pm = [-1,1].\end{gathered} \tag{2.6}$$

- La seconde forme affine AF2 est basée sur AF1. Le nombre de variables y est aussi fixé, mais l'erreur est stockée dans trois termes distincts : l'erreur de signe positif, de signe négatif et de signe indéterminé.

$$\begin{gathered}\widehat{x} = x_0 + \sum_{i=1}^{n} x_i\epsilon_i + x_\pm\epsilon_\pm + x_+\epsilon_+ + x_-\epsilon_-, \\ \text{avec } \forall i \in \{0,1,\ldots,n\}, x_i \in \mathbb{R} \text{ et } \forall i \in \{1,2,\ldots,n\},\ \epsilon_i = [-1,1], \\ \text{et } (x_\pm, x_+, x_-) \in \mathbb{R}^3_+,\ \epsilon_\pm = [-1,1], \epsilon_+ = [0,1], \epsilon_- = [-1,0].\end{gathered} \tag{2.7}$$

Remarque 2.2.1 *Pour simplifier les notations de certaines équations utilisant les formes affines AF1 ou AF2, nous remplacerons $x_\pm$ par x_{n+1}, x_+ par x_{n+2} et x_- par x_{n+3}.*

Les opérateurs et fonctions usuelles sont redéfinis par extension de l'arithmétique affine, pour les détails voir [69]. Par exemple, la multiplication de deux formes AF1 est effectuée de la manière suivante :

$$\widehat{x} \times \widehat{y} = x_0y_0 + \sum_{i=1}^{n}(x_0y_i + x_iy_0)\epsilon_i + \left(|x_0|y_\pm + x_\pm|y_0| + \left(\sum_{i=1}^{n+1}|x_i| \times \sum_{i=1}^{n+1}|y_i|\right)\right)\epsilon_\pm.$$

Pour la multiplication de deux formes AF2, la formulation est un peu plus longue, mais le principe reste le même [1] :

$$\widehat{x} \times \widehat{y} = x_0 y_0 + \sum_{i=1}^{n} (x_0 y_i + x_i y_0)\epsilon_i + K_1 \epsilon_{\pm} + K_2 \epsilon_{+} + K_3 \epsilon_{-}.$$

avec K_1, K_2 et K_3 calculés comme suit :

$$\begin{aligned}
K_1 = \;& |x_0| y_{\pm} + |y_0| x_{\pm} + x_{\pm} y_{\pm} + \sum_{\substack{1 \le i,j \le n+3 \\ i \neq j}} |x_i y_j|, \\
K_2 = \;& K_2^{\{0\}} + x_{+} y_{+} + x_{-} y_{-} + \sum_{\substack{i=1 \\ x_i y_i > 0}}^{n} x_i y_i, \\
K_3 = \;& K_3^{\{0\}} - \sum_{\substack{i=1 \\ x_i y_i < 0}}^{n} x_i y_i,
\end{aligned}$$

où

$$K_2^{\{0\}} = \begin{cases} x_0 y_{+} + y_0 x_{+} & \text{si } x_0 \ge 0 \text{ et } y_0 \ge 0, \\ x_0 y_{+} - y_0 x_{-} & \text{si } x_0 \ge 0 \text{ et } y_0 < 0, \\ -x_0 y_{-} + y_0 x_{+} & \text{si } x_0 < 0 \text{ et } y_0 \ge 0, \\ -x_0 y_{-} - y_0 x_{-} & \text{si } x_0 < 0 \text{ et } y_0 < 0, \end{cases}$$

$$K_3^{\{0\}} = \begin{cases} x_0 y_{-} + y_0 x_{-} & \text{si } x_0 \ge 0 \text{ et } y_0 \ge 0, \\ x_0 y_{-} - y_0 x_{+} & \text{si } x_0 \ge 0 \text{ et } y_0 < 0, \\ -x_0 y_{+} + y_0 x_{-} & \text{si } x_0 < 0 \text{ et } y_0 \ge 0, \\ -x_0 y_{+} - y_0 x_{+} & \text{si } x_0 < 0 \text{ et } y_0 < 0. \end{cases}$$

Tout comme dans la version développée par De Figueiredo et Stolfi [30], pour calculer les fonctions unaires, nous avons choisi l'approximation min-range. Pour une forme affine AF1 ou AF2, les linéarisations sont de la forme de l'équation (2.8).

$$\begin{gathered} \widehat{f}(\widehat{x}) = \zeta + \alpha \widehat{x} + \delta \epsilon_{\pm}, \\ \text{avec } \widehat{x} \text{ donné par l'équation (2.6) ou (2.7) et } \zeta \in \mathbb{R}, \alpha \in \mathbb{R}^{+}, \delta \in \mathbb{R}^{+}. \end{gathered} \tag{2.8}$$

Les algorithmes de linéarisation des opérateurs $\div$, log, exp et $\sqrt{\ }$ consistent donc à rechercher les trois constantes ζ, α et δ de l'équation (2.8).

Exemple 2.2.2 *Considérons la fonction suivante :*

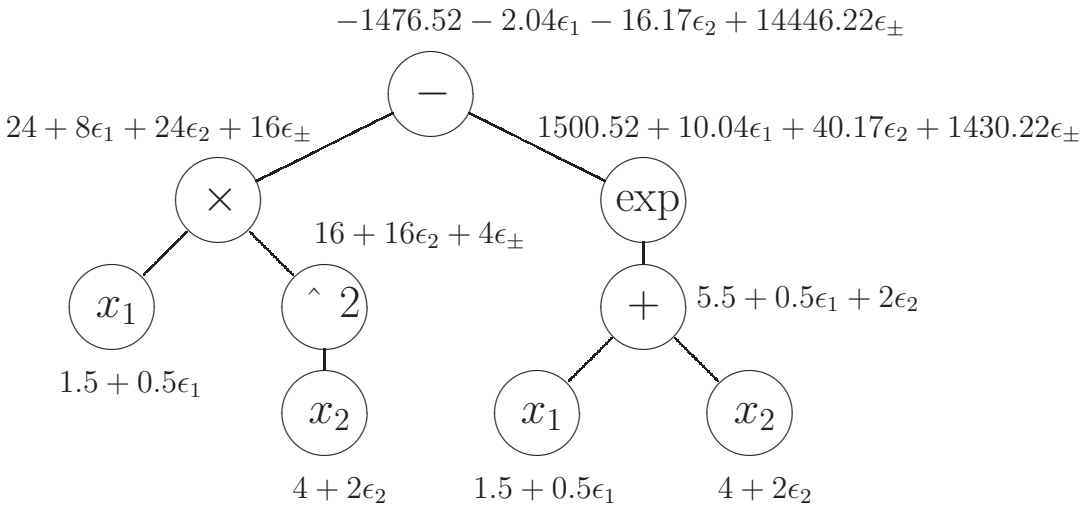
$$\forall x \in [1,2] \times [2,6], \;\; f(x) = x_1 x_2^2 - \exp(x_1 + x_2).$$

Tout d'abord, en utilisant l'équation (2.2), nous transformons les intervalles de chaque variable $\mathbf{x}$ *en forme affine (à ce stade, les formes AF1 et AF2 sont équivalentes).*

$$\mathbf{x}_1 = [1,2] \rightarrow \widehat{x}_1 = 1.5 + 0.5\epsilon_1 \quad et \quad \mathbf{x}_2 = [2,6] \rightarrow \widehat{x}_2 = 4 + 2\epsilon_2.$$

1. Merci à Xuan-Ha Vu, Djamila Sam-Haroud et Boi Faltings [107] pour avoir corrigé une petite erreur dans la formulation de la multiplication de deux formes affines AF2 de l'article [69].

FIGURE 2.2 – Visualisation du calcul avec AF1 sur l'arbre de calcul.
$\forall x \in [1,2] \times [2,6], f(x) = x_1 x_2^2 - \exp(x_1 + x_2)$.



Après calcul en utilisant les opérateurs étendus à AF1 et AF2, on obtient les deux formes affines suivantes :

$$\widehat{f}_{AF1}(x) = -1476.521761 - 2.042768\epsilon_1 - 16.171073\epsilon_2 + 1446.222382\epsilon_\pm,$$
$$\widehat{f}_{AF2}(x) = -1476.521761 - 2.042768\epsilon_1 - 16.171073\epsilon_2 + 1440.222382\epsilon_\pm + 6\epsilon_+.$$

Les détails du calcul de AF1 sont représentés dans la figure 2.2. La variable ϵ_1 correspond à x_1, ϵ_2 à x_2 et avec AF1, $\epsilon_\pm$ contient toute l'erreur générée par les opérations non-affines. Avec AF2, $\epsilon_\pm$ contient l'erreur générée par la multiplication et l'exponentielle, et ϵ_+ l'erreur générée par la puissance de 2.

Pour conclure, en utilisant l'équation (2.3), on reconvertit les formes affines obtenues en intervalles pour obtenir les encadrements suivants, pour tout $x \in [1,2] \times [2,6]$:

En utilisant simplement l'extension naturelle aux intervalles,
$$f(x) \in [-2976.9579870417284, 51.91446307681234],$$
Avec AF1, $f(x) \in [-2940.9579870417297, -12.085536923186737]$,
Avec AF2, $f(x) \in [-2934.9579870417297, -12.085536923186737]$,
Et pour les bornes exactes,
$$f(x) \in [-2908.957987041728, -16.085536923187668].$$

Dans cet exemple, les minorants et majorants calculés par l'arithmétique d'intervalles ne sont pas bons. Un majorant positif introduit une ambiguïté sur le signe de f sur $[1,2] \times [2,6]$ alors que celle-ci est clairement négative. Néanmoins, si l'on calcule les minorants et majorants à chaque nœud de l'arbre de la figure 2.2, on peut s'apercevoir que l'arithmétique d'intervalles donne exactement les bornes inférieures et supérieures (cf théorème 1.1.12), excepté à la racine. Cet exemple met donc en évidence l'intérêt des formes AF1 et AF2, ainsi que son utilisation pour le calcul d'encadrement d'une fonction sur un intervalle.

Dans [69], une comparaison empirique entre l'arithmétique d'intervalles et les formes affines AF1 et AF2 a été réalisée sur des polygones générés aléatoirement et la preuve de la proposition suivante est donnée :

Proposition 2.2.3 *Considérons une fonction polynomiale f de $\mathbb{R}^n$ dans $\mathbb{R}$, $\boldsymbol{X}$ un intervalle de $\mathbb{R}^n$ et f_{AF2}, f_{AF1}et f_{AF} les différentes fonctions d'inclusion de f respectivement dans le arithmétiques AF2, AF1 et AF alors :*

$$[\min_{x\in\boldsymbol{X}} f(x), \max_{x\in\boldsymbol{X}} f(x)] \subseteq f_{AF2}(\widehat{x}) \subseteq f_{AF1}(\widehat{x}) = f_{AF}(\widehat{x}). \tag{2.9}$$

Ainsi, l'arithmétique affine est une approche complètement différente par rapport à l'arithmétique d'intervalles, ou à des méthodes basées sur le développement de Taylor qui nécessitent le calcul du gradient. En arithmétique affine, une approximation du gradient est calculée implicitement à chaque nœud de l'arbre de calcul, mais la formulation du gradient n'est absolument pas nécessaire. La comparaison entre l'arithmétique affine et l'arithmétique d'intervalles peut être mise en parallèle avec la comparaison entre la propagation des contraintes basée sur l'arbre de calcul (cf section 1.3.2 et [70]) et celle basée sur des relaxations linéaires [42].

2.3 Extension au cas mixte : AF3

Kolev dans [58] proposa une forme affine beaucoup plus générale que l'arithmétique affine standard. Il introduisit une formulation affine dans laquelle toutes les variables ϵ_i sont incluses dans des intervalles de type $[-\mu_i, \mu_i]$ avec $\mu_i \in \mathbb{R}^+$, et non simplement $[-1, 1]$ comme cela est défini dans l'arithmétique affine standard et les extensions AF1 et AF2.

Sur cette idée, nous avons donc défini une nouvelle extension de l'arithmétique affine, nommée AF3. Celle-ci est basée sur le même principe que AF2, à l'exception que maintenant les variables ϵ_i sont comprises dans l'intervalle $[0, 1]$, et le terme de l'erreur est réduit aux variables ϵ_+ et ϵ_-. La forme affine AF3 est représentée ainsi :

$$\begin{gathered}\widehat{x} = x_0 + \sum_{i=1}^{n} x_i\epsilon_i + x_+\epsilon_+ + x_-\epsilon_-,\\ \text{avec } \forall i \in \{0,\dots,n\}, x_i \in \mathbb{R} \text{ et } \forall i \in \{1,\dots,n\},\ \epsilon_i = [0,1],\\ \text{et } (x_+, x_-) \in \mathbb{R}^3_+,\ \epsilon_+ = [0,1], \epsilon_- = [-1,0].\end{gathered}$$

Remarque 2.3.1 *Pour simplifier les notations de certaines équations utilisant les formes affines AF3, nous considérerons x_+ comme étant x_{n+1} et x_- comme étant x_{n+2}.*

Tout comme AF2, les opérations affines sont effectuées sans approximation. Pour la multiplication, un calcul est nécessaire. Soit $\widehat{x}$ et $\widehat{y}$ deux formes affines AF3 :

$$\widehat{x} \times \widehat{y} = x_0 y_0 + \sum_{i=1}^{n} (x_0 y_i + x_i y_0)\epsilon_i + K_1 \epsilon_+ + K_2 \epsilon_-,$$

avec K_1 et K_2 calculés comme suit :

$$\begin{aligned} K_1 = \quad & K_1^{\{0\}} + x_- y_- + \sum_{\substack{1 \le i,j \le n+1 \\ x_i y_i > 0}} x_i y_j - y_- \sum_{\substack{i=1 \\ x_i < 0}}^{n+1} x_i - x_- \sum_{\substack{i=1 \\ y_i < 0}}^{n+1} y_i, \\ K_2 = \quad & K_2^{\{0\}} - \sum_{\substack{1 \le i,j \le n+1 \\ x_i y_i < 0}} x_i y_j + y_- \sum_{\substack{i=1 \\ x_i > 0}}^{n+1} x_i + x_- \sum_{\substack{i=1 \\ y_i > 0}}^{n+1} y_i, \end{aligned}$$

où

$$K_1^{\{0\}} = \begin{cases} x_0 y_+ + y_0 x_+ & \text{si } x_0 \ge 0 \text{ et } y_0 \ge 0, \\ x_0 y_+ - y_0 x_- & \text{si } x_0 \ge 0 \text{ et } y_0 < 0, \\ -x_0 y_- + y_0 x_+ & \text{si } x_0 < 0 \text{ et } y_0 \ge 0, \\ -x_0 y_- - y_0 x_- & \text{si } x_0 < 0 \text{ et } y_0 < 0, \end{cases}$$

$$K_2^{\{0\}} = \begin{cases} x_0 y_- + y_0 x_- & \text{si } x_0 \ge 0 \text{ et } y_0 \ge 0, \\ x_0 y_- - y_0 x_+ & \text{si } x_0 \ge 0 \text{ et } y_0 < 0, \\ -x_0 y_+ + y_0 x_- & \text{si } x_0 < 0 \text{ et } y_0 \ge 0, \\ -x_0 y_+ - y_0 x_+ & \text{si } x_0 < 0 \text{ et } y_0 < 0. \end{cases}$$

Concernant les opérateurs unaires non affines, le principe reste le même qu'avec les autres extensions de l'arithmétique affine. Nous avons choisi de n'implémenter que la formulation *min-range*, voir équation (2.8) et figure 2.1. Les algorithmes codant ces opérateurs sont légèrement différents de ceux développés pour AF1 et AF2, cela vient simplement du fait que les variables ϵ_i sont maintenant dans $[0, 1]$.

Remarque 2.3.2 *La motivation originelle de cette nouvelle forme affine est issue de la programmation mixte. En effet, les variables binaires appartiennent à* $[0, 1]$*, il pouvait donc être astucieux d'identifier les* ϵ_i *directement avec une variable binaire. Ainsi, aucune transformation ni approximation n'intervenait. Néanmoins, il s'est avéré que AF3 était peu efficace pour les problèmes ayant des variables binaires. En effet, dans ce type de problème, la multiplication entre deux variables binaires intervient assez fréquemment. Or comme on peut le constater dans l'exemple 2.3.3, la forme affine obtenue après la multiplication de deux variables binaires est généralement de bien meilleure qualité avec AF1 (ou AF2) qu'avec AF3, cette dernière ne se réduisant qu'à l'erreur.*

Ceci peut s'expliquer en observant la formulation de la multiplication. On peut y remarquer que, peu importe l'arithmétique, les termes x_0 *et* y_0 *sont essentiels et déterminent la partie linéaire de la forme affine du produit. Or, la constante* x_0 *de la reformulation en AF3 d'une simple variable binaire est nulle. Contrairement à AF1 et AF2, avec lesquelles la constante vaut* 0.5*. C'est pourquoi avec AF3, à la suite d'un produit de deux variables binaires, on obtient une partie linéaire nulle.*

Exemple 2.3.3 *Illustration du problème lié à la reformulation du produit de deux termes binaires. Soit l'équation suivante :*

$$\forall (x, y) \in [0, 1]^2, f(x, y) = x \times y.$$

En reformulant, f, x et y en forme AF3, on obtient :

$$\widehat{x}_{AF3} = \epsilon_1, \qquad \widehat{y}_{AF3} = \epsilon_2,$$

$$\widehat{f}(x, y)_{AF3} = \epsilon_+.$$

En reformulant, f, x et y en forme AF1, on obtient :

$$\widehat{x}_{AF1} = 0.5 + 0.5\epsilon_1, \qquad \widehat{y}_{AF1} = 0.5 + 0.5\epsilon_2,$$

$$\widehat{f}(x, y)_{AF1} = 0.25 + 0.25\epsilon_1 + 0.25\epsilon_2 + 0.25\epsilon_\pm.$$

On peut remarquer que la forme obtenue avec AF1 est de bien meilleure qualité, car la forme AF3 ne garde aucune information linéaire sur la quantité calculée, ce qui fait perdre tout l'intérêt de l'utilisation de l'arithmétique affine.

2.4 Combinaisons de l'arithmétique affine et de l'arithmétique d'intervalles : IA_AF

Comme nous l'avons vu dans la section 1.1.2, l'arithmétique d'intervalles est très performante pour calculer des minorants et des majorants d'une fonction sur un intervalle, lorsque l'expression de la fonction ne contient qu'une seule occurrence de chaque variable. Néanmoins, en pratique, l'arithmétique affine donne de meilleurs minorants que l'arithmétique d'intervalles pour les fonctions plus complexes. L'idée est donc de combiner à chaque étape du calcul l'arithmétique d'intervalles et l'arithmétique affine.

Définissons une structure nommée *IA_AF*. Celle-ci est constituée à la fois d'une forme affine (AF1, AF2 ou AF3) et d'un intervalle. L'intervalle représente le meilleur encadrement de l'image entre celui calculé par l'arithmétique d'intervalles et celui calculé par l'arithmétique affine. La forme affine est calculée par l'arithmétique affine choisie, à l'exception des opérateurs unaires.

Pour linéariser les opérateurs unaires en arithmétique affine classique, il est normalement nécessaire de convertir la forme affine de départ en intervalle pour connaître l'intervalle sur lequel on va calculer les paramètres α, ζ et δ de l'approximation. En utilisant l'arithmétique *IA_AF*, il n'est pas nécessaire d'effectuer cette conversion, étant donné que l'intervalle est directement accessible. Or, cet intervalle est obtenu en prenant les meilleures bornes de l'arithmétique d'intervalles et de l'arithmétique affine. Il est donc nécessairement moins large que l'intervalle obtenu par simple conversion de la forme affine. L'approximation calculée sera donc de meilleure qualité qu'en arithmétique affine classique.

Considérons $\widehat{x}_{IA\_AF}$ et $\widehat{y}_{IA\_AF}$ deux formes IA_AF. Notons $\widehat{x}_{IA}$ et $\widehat{y}_{IA}$ les intervalles de $\widehat{x}_{IA\_AF}$ et $\widehat{y}_{IA\_AF}$, et $\widehat{x}_{AF}$ et $\widehat{y}_{AF}$ leurs formes affines respectives. Notons $convAF$ la fonction de conversion d'une forme affine en intervalle (voir équation (2.3)) :

Voici quelques exemples d'opérateurs usuels pour illustrer le principe :

– L'addition :

$$\widehat{x+y}_{IA\_AF} = \begin{cases} \widehat{x+y}_{AF} = \widehat{x}_{AF} + \widehat{y}_{AF}, \\ \widehat{x+y}_{IA} = (\widehat{x}_{IA} + \widehat{y}_{IA}) \cap \left(convAF(\widehat{x+y}_{AF}) \right). \end{cases}$$

– La multiplication :

$$\widehat{x\times y}_{IA\_AF} = \begin{cases} \widehat{x\times y}_{AF} = \widehat{x}_{AF} \times \widehat{y}_{AF}, \\ \widehat{x\times y}_{IA} = (\widehat{x}_{IA} \times \widehat{y}_{IA}) \cap \left(convAF(\widehat{x\times y}_{AF}) \right). \end{cases}$$

– Les opérateurs unaires usuels. Par exemple, l'exponentielle :

$$\widehat{\exp(x)}_{IA\_AF} = \begin{cases} \widehat{\exp(x)}_{AF} = & \exp(\widehat{x}_{AF}) \text{ en remplaçant } convAF(\widehat{x}_{AF}) \text{ par} \\ & \widehat{x}_{IA} \text{ dans l'algorithme codant l'exponentielle} \\ & \text{de la forme affine,} \\ \widehat{\exp(x)}_{IA} = & (\exp(\widehat{x}_{IA})) \cap \left(convAF(\widehat{\exp(x)}_{AF}) \right). \end{cases}$$

La forme affine de IA_AF est nécessairement plus précise et génère moins d'erreurs que celle obtenue directement par l'arithmétique affine. En effet, l'intervalle $\widehat{x}_{IA}$ est nécessairement inférieur ou égal à $convAF(\widehat{x}_{AF})$, ainsi l'intervalle sur lequel l'approximation est effectuée, est plus petit et donc la linéarisation est nécessairement plus précise que la simple utilisation de l'arithmétique affine.

Proposition 2.4.1 *L'intervalle de la forme* IA_AF *d'une fonction explicite est une fonction d'inclusion.*

Démonstration. Soient deux formes IA_AF, $\widehat{x}_{IA\_AF}$ et $\widehat{y}_{IA\_AF}$. Supposons que les formes affines et les intervalles de $\widehat{x}_{IA\_AF}$ et $\widehat{y}_{IA\_AF}$ sont des fonctions d'inclusion. Montrons que ceci est toujours vrai pour $\widehat{x+y}_{IA\_AF}$ et pour $\widehat{\exp(x)}_{IA\_AF}$.

Par hypothèse, l'image directe de x est incluse dans $convAF(\widehat{x}_{AF})$ et celle de y dans $convAF(\widehat{y}_{AF})$. Donc l'image directe de la somme $x+y$ est incluse dans $convAF(\widehat{x+y}_{AF}) = convAF(\widehat{x}_{AF} + \widehat{y}_{AF})$. De même, par hypothèse, l'image directe de x est incluse dans $\widehat{x}_{IA}$ et celle de y dans $\widehat{y}_{IA}$. Donc l'image directe de la somme $x+y$ est incluse dans $\widehat{x}_{IA} + \widehat{y}_{IA}$. Ainsi, l'image directe de la somme est incluse dans $\widehat{x+y}_{IA}$.

Par hypothèse, l'image directe de x est incluse dans $\widehat{x}_{IA}$. Notons α, ζ et δ les constantes calculées pour linéariser l'exponentiel sur $\widehat{x}_{IA}$, voir équation (2.5). On a alors $\widehat{\exp(x)}_{AF} = \zeta + \alpha\widehat{x}_{AF} + \delta\epsilon_{\pm}$. Or par construction, l'image directe de $\exp(x)$ est incluse dans $\zeta + \alpha\widehat{x}_{IA} + \delta[-1,1]$. Ainsi, elle est aussi incluse dans $convAF(\widehat{\exp(x)}_{AF})$.

Or, elle est également incluse dans $\exp(\widehat{x}_{IA})$. Donc, finalement, l'image directe de $\exp(x)$ est incluse dans $\widehat{\exp(x)}_{IA}$.

Par analogie, aux deux cas précédents, il est possible de prouver que l'image directe reste incluse dans les formes affines et les intervalles des formes *IA_AF* après application de n'importe quels opérateurs ou fonctions de l'arithmétique. Étant donné que cette proposition est vraie pour les fonctions réduites à une simple variable. On peut en déduire que l'image directe d'une fonction explicite est incluse dans l'intervalle de la forme *IA_AF* de cette fonction. Ainsi, la proposition est démontrée.

■

D'après la proposition précédente, il est donc possible d'obtenir un minorant et un majorant d'une fonction sur un intervalle, en considérant l'intervalle final de la forme *IA_AF* de la fonction. Cet encadrement est nécessairement inférieur ou égal à l'encadrement obtenu par l'extension naturelle, car c'est dans le pire des cas l'intersection de l'extension naturelle aux intervalles et de la fonction d'inclusion de l'arithmétique affine. De plus, la forme affine obtenue est de meilleure qualité, puisque les intervalles considérés pour linéariser les fonctions non-affines sont plus précis, ce qui diminue nécessairement l'erreur générée.

De plus, étant donné que le calcul de l'encadrement est contrôlé à la fois par l'arithmétique d'intervalles et par la forme affine, il est intéressant de considérer maintenant les approximations de Chebyshev pour reformuler les fonctions non-affines (voir figure 2.1). En effet, on a pu remarquer dans la partie 2.1 que le calcul de l'encadrement pouvait être beaucoup plus large avec ces approximations que l'image exacte. En revanche, l'erreur générée par l'approximation est minimale. Or, dans la forme *IA_AF*, le calcul de l'encadrement est dans le pire des cas celui de l'arithmétique d'intervalles seul. Il n'est donc plus nécessaire de se focaliser sur le calcul des minorants et majorants. L'arithmétique d'intervalles contrôle l'encadrement et la forme affine peut essayer de garder le maximum de dépendance linéaire entre les variables pour fournir une bonne approximation linéaire de la fonction totale ou pour exploiter ces dépendances lorsqu'il y a des répétitions de variables, dans l'espoir d'améliorer l'encadrement.

Notons *IA_AF1* et *IA_AF2* les formes combinant l'arithmétique d'intervalles et une forme affine AF1 ou respectivement AF2.

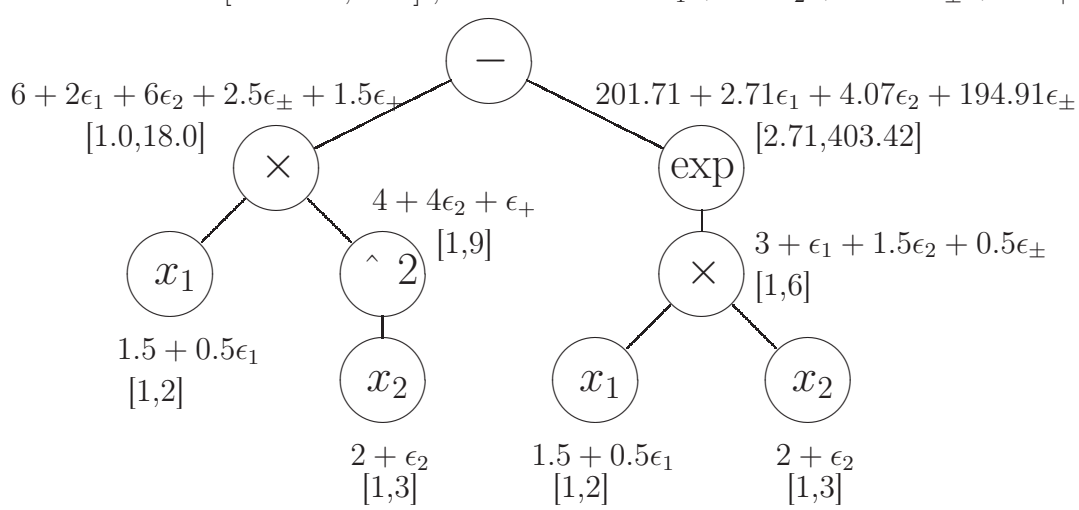
La figure 2.3 représente le comportement de l'arithmétique *IA_AF2* sur l'arbre de calcul, dans le cas où l'approximation min-range est choisie pour linéariser l'exponentielle. En comparant *IA_AF2* avec l'arithmétique d'intervalles et AF2, on obtient les encadrements suivants pour $f(x) = x_1x_2^2 - \exp(x_1x_2)$:

- Avec l'arithmétique d'intervalles,
  $\forall x \in [1,2] \times [1,3], f(x) \in [-402.42879349273517, 15.281718171540956]$.
- Avec l'arithmétique affine AF2 avec linéarisation min-range,
  $\forall x \in [1,2] \times [1,3], f(x) \in [-402.92879349273517, 12.0]$.
- Avec l'arithmétique affine AF2 avec linéarisation de Chebyshev,
  $\forall x \in [1,2] \times [1,3], f(x) \in [-391.92879349273523, 217.01493440574007]$.

– Avec l'arithmétique *IA_AF2* avec linéarisation min-range,
$\forall x \in [1,2] \times [1,3], f(x) \in [-395.77394800735812, 5.8451545146229194]$.
– Avec l'arithmétique *IA_AF2* avec linéarisation de Chebyshev,
$\forall x \in [1,2] \times [1,3], f(x) \in [-391.92879349273517, 15.281718171540956]$.

On remarque notamment que sur ce simple exemple, l'arithmétique d'intervalles et l'arithmétique affine ne sont pas comparables. En revanche, dans *IA_AF2*, les avantages des deux arithmétiques sont combinés pour obtenir un encadrement de meilleure qualité. On remarque notamment que le majorant de la forme AF2 avec linéarisation de Chebyshev est vraiment mauvais alors que celui de *IA_AF2* est égale à celui de l'arithmétique d'intervalles tout en ayant un meilleur minorant que celle-ci.

FIGURE 2.3 – Visualisation du calcul avec IA_AF2 sur l'arbre de calcul.
$\forall x \in [1,2] \times [1,3], f(x) = x_1 x_2^2 - \exp(x_1 x_2)$.



Il est possible de comparer les formes affines obtenues avec *IA_AF2* et AF2.
– Avec AF2 et linéarisation min-range :
$\widehat{f}(x)_{AF2} = -196.21 + \epsilon_1 + 4.5\epsilon_2 + 201.214\epsilon_\pm + 1.5\epsilon_+$.
– Avec AF2 et linéarisation de Chebyshev :
$\widehat{f}(x)_{AF2} = -88.2 - 65.07\epsilon_1 - 94.60\epsilon_2 + 144.04\epsilon_\pm + 1.5\epsilon_+$.
– Avec IA_AF2 et linéarisation min-range :
$\widehat{f}(x)_{IA\_AF2} = -195.71 - 0.71\epsilon_1 + 1.92\epsilon_2 + 197.41\epsilon_\pm + 1.5\epsilon_+$.
– Avec IA_AF2 et linéarisation de Chebyshev :
$\widehat{f}(x)_{IA\_AF2} = -60.1 - 78.14\epsilon_1 + 114.21\epsilon_2 + 139.45\epsilon_\pm + 1.5\epsilon_+$.

On remarque que le simple fait d'avoir amélioré le calcul de l'image pour la linéarisation de l'exponentielle a eu un impact favorable sur la qualité de la forme affine, car pour la même méthode de linéarisation, l'erreur diminue. Néanmoins, on observe très nettement que les coefficients des variables ϵ_i sont beaucoup plus grands avec la méthode de Chebyshev. Il y a donc plus d'information gardée pendant le

calcul, les influences linéaires des variables dans chaque quantité sont beaucoup plus conservées et, de plus, les coefficients de l'erreur y sont plus petits, ce qui veut dire que l'approximation est plus précise.

Pour évaluer la qualité des formes affines, on peut observer sur la figure 2.4 une coupe représentant la fonction $f(x) = x_1 x_2^2 - \exp(x_1 x_2)$. Cette coupe est réalisée suivant l'équation $x_1 = 1.5x_2$ (c'est-à-dire pour $\epsilon_1 = \epsilon_2$). On y distingue également toutes les formes affines précédentes moins l'erreur inférieure maximale, ce qui implique que la fonction f est nécessairement au-dessus de toutes les droites. On peut donc remarquer que les formulations utilisant Chebyshev donnent des approximations linéaires de meilleure qualité, en minimisant l'erreur.

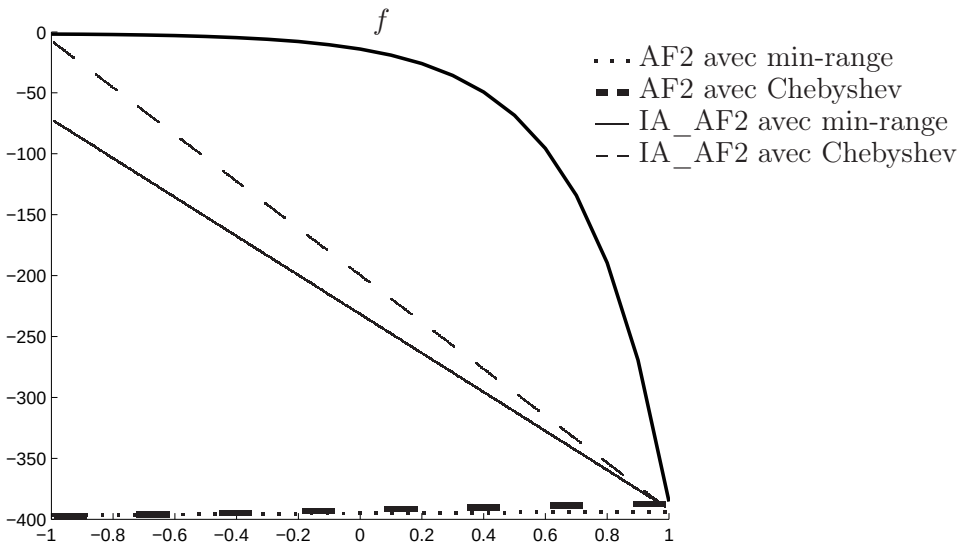


FIGURE 2.4 – Représentation de l'encadrement inférieur des différentes formes affines de l'équation $f(x) = x_1 x_2^2 - \exp(x_1 x_2)$ sur $[1, 2] \times [1, 3]$ pour la coupe d'équation $x_1 = 1.5x_2$.

Ainsi, la formulation *IA_AF2* avec linéarisation de Chebyshev apparaît comme un bon compromis entre qualité des minorants et qualité de la linéarisation.

2.5 Robustesse des arithmétiques affines

Tout comme l'arithmétique d'intervalles, les arithmétiques affines sont utilisées pour calculer des encadrements de fonctions sur des intervalles. Néanmoins, le caractère robuste et fiable du calcul de ces encadrements n'est pas aussi simple à réaliser que pour l'arithmétique d'intervalles.

Nous allons donc décrire trois principes distincts pour rendre celles-ci complètement fiables et robustes aux erreurs numériques. Les formules et les équations restent les mêmes, seule la façon d'effectuer les calculs et de prendre en compte les erreurs numériques change.

2.5.1 Self-Validated Affine Arithmetic : *sAF*

Stolfi et De Figueiredo proposent dans [101] une version fiable et robuste de l'arithmétique affine, nommée *self-validated affine arithmetic*, qui peut facilement être adaptée et étendue aux formes AF1, AF2 et AF3.

Cette version consiste à calculer l'erreur d'approximation maximale à chaque calcul des quantités intermédiaires de l'équation. Chaque quantité est donc calculée en mode d'arrondi standard (c'est-à-dire au nombre flottant le plus proche), en mode d'arrondi au nombre flottant inférieur le plus proche, puis en mode d'arrondi au nombre flottant supérieur le plus proche. Au final, on évalue l'écart maximal entre la valeur obtenue par le calcul standard et les deux autres. Cet écart est ensuite pris en compte dans une nouvelle variable (pour l'arithmétique affine standard), ou additionné dans le terme $\epsilon_{\pm}$ (pour AF1, AF2 et AF3).

Ce calcul peut s'effectuer en utilisant la fonction NEAREST en Fortran ou par changement du mode d'arrondi du processeur dans d'autres langages tels que le C. L'algorithme 4 décrit la technique utilisée pour le calcul de la fonction suivant : $\widehat{z} = \zeta + \alpha\widehat{x} + \widehat{y} + \delta\epsilon_{\pm}$, avec $\widehat{x}$, $\widehat{y}$ et $\widehat{z}$ des formes affines AF2 et α, ζ et δ des constantes. Les flèches $\uparrow a + b \uparrow$ indiquent que le calcul $a + b$ est effectué en l'arrondissant au nombre flottant supérieur, et $\downarrow a + b \downarrow$ au nombre flottant inférieur. Tous les opérateurs classiques des extensions de l'arithmétique affine peuvent être ensuite déduits à partir de cet algorithme. Le détail de chacune des opérations est isolé afin de faciliter la compréhension.

Malheureusement, le changement du mode d'arrondi d'un processeur est une opération très coûteuse en temps. En effet, à chaque changement, toutes les opérations mises en cache doivent être vidées, ce qui peut représenter un temps d'attente allant jusqu'à 60 cycles, alors qu'une opération flottante ne prend qu'un ou deux cycles. Les performances de l'arithmétique risquent donc de diminuer sensiblement.

2.5.2 reliable Affine Arithmetic : *rAF*

Dans [103], Touhami et Messine proposent une version des formes AF1 et AF2 complètement robuste et fiable, nommée *reliable Affine Arithmetic* pour *arithmétique affine robuste*. Le principe consiste à remplacer dans la forme affine tous les

Algorithme 4 Calcul de la fonction $\widehat{z} = \zeta + \alpha\widehat{x} + \widehat{y} + \delta\epsilon_{\pm}$ en *self-validated affine arithmetic sAF2*

1: {Calcul de $\widehat{z} = \alpha\widehat{x}$}
2: $error = 0$
3: **pour** $i = 0, n$ **faire**
4: $\quad z_i = \alpha x_i$
5: $\quad error = \uparrow error + \max(\uparrow \alpha x_i \uparrow - z_i, z_i - \downarrow \alpha x_i \downarrow) \uparrow$
6: **fin pour**
7: **si** $\alpha > 0$ **alors**
8: $\quad z_{\pm} = \uparrow \alpha x_{\pm} + error \uparrow$
9: $\quad z_{+} = \uparrow \alpha x_{+} \uparrow$
10: $\quad z_{-} = \uparrow \alpha x_{-} \uparrow$
11: **sinon**
12: $\quad z_{\pm} = \uparrow \alpha x_{\pm} + error \uparrow$
13: $\quad z_{+} = \uparrow \alpha x_{-} \uparrow$
14: $\quad z_{-} = \uparrow \alpha x_{+} \uparrow$
15: **finsi**
16: {Calcul de $\widehat{z} = \widehat{z} + \widehat{y}$}
17: $error = 0$
18: **pour** $i = 0, n$ **faire**
19: $\quad tmp = z_i + y_i$
20: $\quad error = \uparrow error + \max(\uparrow z_i + y_i \uparrow - tmp, tmp - \downarrow z_i + y_i \downarrow) \uparrow$
21: $\quad z_i = tmp$
22: **fin pour**
23: $z_{\pm} = \uparrow z_{\pm} + y_{\pm} + error \uparrow$
24: $z_{+} = \uparrow z_{+} + y_{+} \uparrow$
25: $z_{-} = \uparrow z_{-} + y_{-} \uparrow$
26: {Calcul de $\widehat{z} = \zeta + \widehat{z}$}
27: $error = 0$
28: $tmp = z_0 + \zeta$
29: $error = \uparrow error + \max(\uparrow z_0 + \zeta \uparrow - tmp, tmp - \downarrow z_0 + \zeta \downarrow) \uparrow$
30: $z_0 = tmp$
31: $z_{\pm} = \uparrow z_{\pm} + error \uparrow$
32: {Calcul de $\widehat{z} = \widehat{z} + \delta\epsilon_{\pm}$}
33: $z_{\pm} = \uparrow z_{\pm} + \delta \uparrow$

nombres flottants par des intervalles, produisant ainsi une forme affine d'intervalles, voir équation (2.10).

$$\widehat{\mathbf{x}} = \mathbf{x}_0 + \sum_{i=1}^{n} \mathbf{x}_i \epsilon_i, \quad \text{avec } \forall i \in \{0, \ldots, n\}, \mathbf{x}_i = [\underline{\mathbf{x}_i}, \overline{\mathbf{x}_i}] \in \mathbb{I} \text{ et } \forall i \in \{1, \ldots, n\}, \epsilon_i = [-1, 1]. \tag{2.10}$$

Cette arithmétique reste compatible avec l'arithmétique d'intervalles et les arithmétiques affines. Les conversions entre elles se réalisent comme l'indique le tableau 2.1.

TABLE 2.1 – Conversions entre les différentes arithmétiques.

Reliable Affine Form $\longrightarrow$ Interval

$$\widehat{\mathbf{x}} = \mathbf{x_0} + \sum_{i=1}^{n} \mathbf{x_i} \epsilon_i, \longrightarrow$$
$$\mathbf{x} = \mathbf{x_0} + \left(\sum_{i=1}^{n} (\mathbf{x_i} \times [-1, 1]) \right).$$

Affine Form $\longrightarrow$ Reliable Affine Form

$$\widehat{x} = x_0 + \sum_{i=1}^{n} x_i \epsilon_i, \longrightarrow$$
$$\forall i \in \{1, \ldots, n\}, \ \mathbf{x_i} = x_i,$$
$$\widehat{\mathbf{x}} = \mathbf{x_0} + \sum_{i=1}^{n} \mathbf{x_i} \epsilon_i.$$

Interval $\longrightarrow$ Reliable Affine Form

$$\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}], \longrightarrow$$
$$\mathbf{x_0} = mid(\mathbf{x})$$
$$\widehat{\mathbf{x}} = \mathbf{x_0} + \max(\overline{\mathbf{x_0} - \mathbf{x}}, \overline{\mathbf{x} - \mathbf{x_0}}) \epsilon_k,$$
avec ϵ_k une nouvelle variable.

Reliable Affine Form $\longrightarrow$ Affine Form

$$\widehat{\mathbf{x}} = \mathbf{x_0} + \sum_{i=1}^{n} \mathbf{x_i} \epsilon_i, \longrightarrow$$
$$\forall i \in \{1, \ldots, n\}, x_i = mid(\mathbf{x_i}),$$
$$\widehat{x} = x_0 + \sum_{i=1}^{n} x_i \epsilon_i + \left(\sum_{i=1}^{n} \max(\overline{x_i - \mathbf{x_i}}, \overline{\mathbf{x_i} - x_i}) \right) \epsilon_{\pm}.$$

Dans *l'arithmétique affine robuste*, toutes les opérations affines sont réalisées de la même façon que l'arithmétique affine standard, excepté que dans cette version, les calculs s'effectuent en remplaçant l'arithmétique flottante par l'arithmétique d'intervalles pour assurer leur fiabilité. Dans [103], seule la multiplication est explicitée. Pour les autres opérateurs non-affines, nous avons réutilisé le même principe de linéarisation que l'arithmétique affine standard (voir équation (2.5) et (2.8)).

L'algorithme 5 est une généralisation de la linéarisation min-range pour trouver la reformulation de toutes les fonctions usuelles continues, monotones et convexes ou concaves de *l'arithmétique affine robuste*. Tout comme dans les algorithmes de De Figueiredo et Stolfi, l'algorithme 5 consiste à rechercher, de manière fiable, les trois scalaires α, ζ et δ des équations (2.5) et (2.8) et de la figure 2.1.

Remarque 2.5.1 *Cette arithmétique est un cas particulier de* l'arithmétique d'intervalles généralisée *introduit par Hansen dans [41]. Celle-ci est équivalente à une*

Algorithme 5 Linéarisation Min-range de f sur $\widehat{\mathbf{x}}$ pour *l'arithmétique affine robuste rAF2*

1: Toutes les variables locales sont des intervalles : $\mathbf{d}$, $\boldsymbol{\alpha}$, $\boldsymbol{\zeta}$ et $\boldsymbol{\delta}$. Soit F = une fonction d'inclusion de f, F' = une fonction d'inclusion du gradient de f, $\widehat{\mathbf{x}}$ = la *reliable affine form* étudiée, $\mathbf{x}$ = l'intervalles correspondant à la conversion de $\widehat{\mathbf{x}}$ (voir tableau 2.1),

2: $$\begin{cases} \boldsymbol{\alpha} = \underline{F'(\mathbf{x})}, & \mathbf{d} = \left[\underline{F(\mathbf{x}) - \boldsymbol{\alpha} \times \underline{\mathbf{x}}},\ \overline{F(\mathbf{x}) - \boldsymbol{\alpha} \times \overline{\mathbf{x}}}\right], \text{ si } f \text{ est croissant,} \\ \boldsymbol{\alpha} = \overline{F'(\mathbf{x})}, & \mathbf{d} = \left[\underline{F(\mathbf{x}) - \boldsymbol{\alpha} \times \overline{\mathbf{x}}},\ \overline{F(\mathbf{x}) - \boldsymbol{\alpha} \times \underline{\mathbf{x}}}\right], \text{ si } f \text{ est décroissante,} \\ \boldsymbol{\alpha} = 0, & \mathbf{d} = \mathbf{x}, \text{ si } f \text{ est constante} \end{cases}$$

3: $\boldsymbol{\zeta} = mid(\mathbf{d})$,

4: $\boldsymbol{\delta} = \max\left(\overline{\boldsymbol{\zeta} - \mathbf{d}}, \overline{\mathbf{d} - \underline{\boldsymbol{\zeta}}}\right)$,

5: $\widehat{\mathbf{f}}(\widehat{\mathbf{x}}) = \boldsymbol{\zeta} + \boldsymbol{\alpha} \times \widehat{\mathbf{x}} + \boldsymbol{\delta}\epsilon_{\pm}$ {*la reliable affine form* de f sur $\widehat{\mathbf{x}}$}

forme affine avec des intervalles comme coefficients. La multiplication a la même définition que l'arithmétique affine robuste. *Néanmoins, la division ne se généralise pas facilement et de l'information affine est perdue. De plus, rien n'est défini dans [41] pour les fonctions non-affines telles que le logarithme, l'exponentielle et la racine carrée.*

Cette arithmétique est complètement fiable : elle gère implicitement les approximations liées à la précision machine ainsi que les erreurs d'arrondies qui sont entièrement prises en compte par l'arithmétique d'intervalles. Toutefois, il convient de remarquer que considérer des intervalles à la place de nombres flottants peut avoir un impact significatif sur les performances.

2.5.3 Floating-point Affine Arithmetic : *fAF*

Cette troisième version de l'arithmétique affine est inspirée de la technique utilisée dans [93] pour prouver la robustesse de l'arithmétique basée sur des méthodes de Taylor intégrées dans l'algorithme *COSY*.

Dans [93], Revol et al. utilisent uniquement l'arithmétique flottante IEEE-754 avec le mode d'arrondi standard [2], c'est-à-dire que les résultats des calculs sont arrondis au nombre flottant le plus proche. Un majorant de toutes les erreurs d'arrondi ainsi générées est ensuite évalué sans changer le mode d'arrondi du processeur, puis additionné dans le terme de l'erreur de la forme affine AF1 ou AF2. Ce majorant est calculé grâce à l'erreur de troncature maximale, c'est-à-dire 2^{-23} en simple précision ou 2^{-52} en double précision.

Les algorithmes 6 et 7 décrivent le calcul de la fonction $\widehat{z} = \zeta + \alpha\widehat{x} + \widehat{y} + \delta\epsilon_{\pm}$ en utilisant le principe précédent et les formes affines AF2. Toutes les constantes des algorithmes sont des nombres flottants double précision. La variable e_M désigne un majorant de l'erreur de troncature maximal de l'arithmétique flottante et e_c le seuil de coupure en dessous duquel le résultat est remplacé par 0. Pour la double

Algorithme 6 Calcul de la fonction $\widehat{z} = \alpha\widehat{x}$ en *floating-point affine arithmetic fAF2*

$t = 0$, $s = 0$
pour i=0,n **faire**
  $z_i = \alpha x_i$
  $t = t + |z_i|$
  si $|z_i| < e_c$ **alors**
    $s = s + |z_i|$
    $z_i = 0$
  finsi
fin pour
si $\alpha > 0$ **alors**
  pour i=n+1,n+3 **faire**
    $z_i = \alpha x_i$
    $t = t + |z_i|$
    si $|z_i| < e_c$ **alors**
      $s = s + |z_i|$
      $z_i = 0$
    finsi
  fin pour
sinon
  $z_\pm = -\alpha x_\pm$
  $t = t + |z_\pm|$
  si $|z_\pm| < e_c$ **alors**
    $s = s + |z_\pm|$
    $z_\pm = 0$
  finsi
  $z_+ = -\alpha x_-$
  $t = t + |z_+|$
  si $|z_+| < e_c$ **alors**
    $s = s + |z_+|$
    $z_+ = 0$
  finsi
  $z_- = -\alpha x_+$
  $t = t + |z_-|$
  si $|z_-| < e_c$ **alors**
    $s = s + |z_-|$
    $z_- = 0$
  finsi
finsi
$z_\pm = \uparrow z_\pm + 2(e_M t) + 2s\uparrow$

Algorithme 7 Calcul de la fonction $\widehat{z} = \zeta + \widehat{x} + \widehat{y} + \delta\epsilon_{\pm}$ en *floating-point affine arithmetic fAF2*

1: {Calcul de $\widehat{z} = \widehat{x} + \widehat{y}$}
2: $t = 0$, $s = 0$
3: **pour** i=0,n+3 **faire**
4: $z_i = x_i + y_i$
5: $t = t + \max(|x_i|, |y_i|)$
6: **si** $|z_i| < e_c$ **alors**
7: $s = s + |z_i|$
8: $z_i = 0$
9: **finsi**
10: **fin pour**
11: $z_{\pm} =\uparrow z_{\pm} + 2(e_M t) + 2s\uparrow$
12: {Calcul de $\widehat{z} = \zeta + \widehat{z}$}
13: $t = 0$, $s = 0$
14: $tmp = z_0 + \zeta$
15: $t = \max(|z_0|, |\zeta|)$
16: **si** $|tmp| < e_c$ **alors**
17: $s = s + |tmp|$
18: $z_0 = 0$
19: **sinon**
20: $z_0 = tmp$
21: **finsi**
22: $z_{\pm} =\uparrow z_{\pm} + 2(e_M t) + 2s\uparrow$
23: {Calcul de $\widehat{z} = \widehat{z} + \delta\epsilon_{\pm}$}
24: $t = 0$, $s = 0$
25: $tmp = z_{\pm} + \delta$
26: $t = \max(|z_{\pm}|, |\delta|)$
27: **si** $|tmp| < e_c$ **alors**
28: $s = s + |tmp|$
29: $z_{\pm} = 0$
30: **sinon**
31: $z_{\pm} = tmp$
32: **finsi**
33: $z_{\pm} =\uparrow z_{\pm} + 2(e_M t) + 2s\uparrow$

précision nous prendrons : $e_M = 2^{-51}$ et $e_c = 2^{-55}$. La preuve de cet algorithme est identique à celle de l'article [93], seules quelques adaptations mineures doivent être effectuées.

2.6 Comparaisons empiriques des nouvelles arithmétiques affines

Nous avons effectué une comparaison de toutes les différentes arithmétiques affines qui ont été développées dans ce chapitre. Le test consiste à effectuer 10000 évaluations de la fonction Γ_{em} décrite dans le chapitre 5 sur le domaine $X = [0.135, 0.14] \times [0.05, 0.055] \times [0.001] \times [0.004, 0.006] \times [0.005, 0.008] \times [0.004, 0.006] \times [0.85] \times [3300000, 3800000] \times [0.004, 0.0045] \times [0.004, 0.0055] \times [130] \times [55] \times [50]$, $Z = [8] \times [3] \times [1]$, $B = [0] \times [1]$ et $K = [1] \times [1]$. Cette fonction est extrêmement non convexe, son expression est assez complexe. Elle contient des divisions, des racines carrées et des logarithmes et les combine de façon non linéaire. Cette fonction permet donc d'avoir une bonne opinion des performances de chaque arithmétique pour évaluer des fonctions longues et complexes. Pour chaque arithmétique, le temps de calcul 't' de 10000 évaluations et l'intervalle résultat sont reportés dans le tableau 2.2, c'est-à-dire qu'on effectue 10000 fois la même évaluation pour obtenir des temps de calcul supérieurs à 1 seconde.

Les arithmétiques comparées sont les suivantes :

- ***IA :*** l'arithmétique d'intervalles arrondie,
- ***AF2 :*** l'extension de l'arithmétique affine développé par Messine [69] et décrite dans la section 2.2, avec la linéarisation min-range,
- ***sAF2 :*** *self-validated affine arithmetic* développée par De Figueiredo et Solfi [101], décrite dans la section 2.5.1 et basée sur l'arithmétique AF2, avec la linéarisation min-range,
- ***rAF2 :*** *reliable affine arithmetic* développé par Messine et Touhami [103], décrite dans la section 2.5.2 et basée sur l'arithmétique AF2, avec la linéarisation min-range,
- ***fAF2 :*** *floating-point affine arithmetic* inspiré de COSY [93], décrite dans la section 2.5.3 et basée sur l'arithmétique AF2, avec la linéarisation min-range,
- ***AF3 :*** l'extension de l'arithmétique affine introduite dans la section 2.3, avec la linéarisation min-range,
- ***rAF3 :*** *reliable affine arithmetic* basée sur l'arithmétique AF3, avec la linéarisation min-range,
- ***IA_AF2 :*** la combinaison de l'arithmétique d'intervalles et l'arithmétique affine AF2, selon le principe décrit dans la section 2.4, avec la linéarisation de Chebyshev dans AF2,
- ***IA_AF3 :*** la combinaison de l'arithmétique d'intervalles et l'arithmétique affine AF3, avec la linéarisation de Chebyshev,
- ***IA_rAF2 :*** la combinaison de l'arithmétique d'intervalles et l'arithmétique affine robuste rAF2, avec la linéarisation de Chebyshev,

– ***IA_rAF3 :*** la combinaison de l'arithmétique d'intervalles et l'arithmétique affine robuste rAF3, avec la linéarisation de Chebyshev.

Avant tout, on peut préciser que les résultats obtenus avec les arithmétiques *AF2*, *AF3*, *IA_AF2* et *IA_AF3* ne sont pas exactement rigoureux, c'est-à-dire qu'elles ne prennent pas en compte l'erreur d'arrondie liée à la précision numérique.

Arithmétique	t (s)	Intervalle
IA	0.106	[1.1832346840604985,23.181133562472031]
AF2	0.985	[-0.088938788418078591,9.6251259942117802]
AF3	0.995	[0.061760629453324167,10.750835220205831]
sAF2	95.380	[-0.31212657498333369,9.8547543783870105]
rAF2	37.999	[-0.31212657498288099,9.8547543783864793]
fAF2	1.928	[-0.31212657498714303,9.8547543783917427]
rAF3	31.222	[0.061760629453303739,10.750835220205943]
IA_AF2	2.944	[2.3121670536079754,8.2495791880595118]
IA_AF3	2.897	[1.5977525441327824,8.7589270934141279]
IA_rAF2	40.966	[2.219897777853653,8.2672007676965205]
IA_rAF3	33.891	[1.5977525441327693,8.7589270934142807]

TABLE 2.2 – Comparaisons des nouvelles arithmétiques affines sur 10000 évaluations de la fonction Γ_{em}.

D'après le tableau 2.2, on observe :

– Pour effectuer les 10000 évaluations, l'arithmétique la plus rapide est l'arithmétique d'intervalles, ce qui peut s'expliquer par son principe simple et sa structure légère. Les arithmétiques affines *AF2* et *AF3* ne nécessitent que 0.9s, les combinaisons *IA_AF2* et *IA_AF3* nécessitent 2.9s. La structure *IA_AF* multiplie donc par **3** le temps de calcul par rapport à l'utilisation de l'arithmétique affine seule.
– Les trois versions robustes *sAF2*, *rAF2* et *fAF2* ont des temps de calcul très différents pour des résultats équivalents à 10^{-10} près. *fAF2* divise les temps de calcul par **18** par rapport à *rAF2* et par **47** par rapport à *sAF2*. Ainsi, on peut remarquer que la façon d'implémenter la robustesse a un gros impact sur les performances. La minimisation du nombre de changements de mode d'arrondi est nécessaire pour obtenir des temps de calcul convenables.
– La différence des temps de calcul entre les arithmétiques affines *AF* et les combinaisons avec l'arithmétique d'intervalles *IA_rAF* est d'environ 2s, ce qui doit correspondre au temps de calcul nécessaire pour gérer la structure d'arbre et effectuer les calculs de l'arithmétique d'intervalles.
– La largeur des intervalles des arithmétiques affines est inférieure à celle de l'arithmétique d'intervalles.
– La qualité des intervalles est bien meilleure avec les combinaisons d'arithmétiques *IA_AF*.

– En considérant les intervalles résultats, il existe un écart non négligeable entre les versions robustes et non robustes. Cela montre le réel intérêt de prendre en considération les erreurs d'arrondis, car dans ce type d'équation, il peuvent être propagées et mener à des résultats aberrants.

2.7 Discussion

Dans ce chapitre, nous avons étudié différentes techniques basées sur le principe de l'arithmétique affine.

L'arithmétique affine AF2 est l'extension ayant obtenu les meilleurs résultats, car d'après le théorème 2.2.3 l'arithmétique AF2 donne des meilleurs encadrements que l'arithmétique AF1. De plus, d'après la remarque 2.3.2, l'arithmétique AF3 est à déconseiller si la fonction contient des variables binaires, ce qui limite beaucoup son utilisation.

Les combinaisons IA_AF de la section 2.4 donnent de très bons encadrements bien que la structure soit peut-être encore un peu lourde.

Parmi les trois façons de coder les arithmétiques affines robustes de la section 2.5, la méthode *floating-point affine arithmetic* est la plus efficace en divisant les temps de calcul par **20**. Néanmoins par la suite nous avons privilégié la technique *reliable affine arithmetic*, car son implémentation est beaucoup plus simple puisqu'elle est entièrement basée sur l'arithmétique d'intervalles.

Chapitre 3

Méthode de Reformulation Affine basée sur l'Arithmétique Affine : ART_{AF}

Comme nous l'avons vu dans le chapitre 1, les algorithmes de Branch and Bound par intervalles ont pu bénéficier d'avancées provenant de beaucoup de domaines différents, et grâce à son principe très général, ils ont pu être appliqués à la résolution de problèmes provenant de domaines très diversifiés : chimie, génie électrique, géométrie, etc. Une autre approche pour résoudre les problèmes d'optimisation globale, qui fut développée initialement de façon totalement dissociée des techniques de Branch and Bound par intervalles, est la Technique de Linéarisation-Reformulation, nommée *Reformulation-Linearization Technique (RLT)*. Elle fut introduite par Adams et Sherali [96], voir aussi [8, 88] pour la résolution de problèmes quadratiques et [98] pour le cas polynomial. L'idée de cette approche consiste à reformuler un problème d'optimisation globale en un problème linéaire de plus grande taille en ajoutant de nouvelles variables et des contraintes linéaires pour remplacer les fonctions non linéaires.

Kearfott [53], Kearfott et Hongthong [54] et Lebbah, Rueher et Michel [61, 62] ont inclus ce type de relaxation dans des algorithmes de Branch and Bound par intervalles, montrant ainsi une certaine efficacité sur des exemples numériques. Néanmoins, il n'est pas rare que la résolution du problème linéaire généré requiert beaucoup de temps à chaque itération, notamment à cause de sa grande taille. Ainsi, si le problème a beaucoup de termes non linéaires, cette méthode va générer beaucoup de nouvelles variables ainsi que de nouvelles contraintes.

L'idée que nous proposons est de contourner ce problème en utilisant l'arithmétique affine (cf. chapitre 2) comme une technique de relaxation automatique. Ce chapitre reprend dans un premier temps la technique publiée dans [86], ainsi que l'extension décrite dans [84].

3.1 Principe de la relaxation affine : *ART*

Les techniques de relaxation linéaire n'ont cessé de prendre de l'importance en optimisation globale depuis plusieurs années [4, 8, 46, 54, 61, 64, 65, 88, 96, 102]. Le principe consiste à approximer un problème général d'optimisation par un programme linéaire. Ainsi, la résolution du problème linéaire fournit des bornes ou des certificats d'infaisabilité sur le problème original. La nouveauté de notre approche réside dans la façon dont sont réalisées ces relaxations linéaires.

Avec notre méthode, nommée *Technique de Reformulation Affine* ou *Affine Reformulation Technique* (ART), nous conservons l'arbre de calcul que nous combinons aux extensions de l'arithmétique affine (AF1 ou AF2). En effet, les extensions de l'arithmétique affine manipulent des formes affines sur des arbres de calcul. Mais jusqu'à maintenant, celles-ci n'étaient utilisées que pour calculer des encadrements de fonctions. Ici, notre approche utilise les extensions de l'arithmétique affine, non pas comme un simple moyen de calculer des encadrements, mais comme un moyen de linéariser automatiquement toutes les fonctions possédant une *expression explicite*. Ceci devient possible grâce à la fixation du nombre de variables ϵ_i introduites dans AF1 et AF2. Ainsi, il existe une transformation affine $\mathcal{T}$ entre l'intervalle de départ $\mathbf{X} \subset \mathbb{R}^n$ et $\varepsilon = [-1, 1]^n$, voir l'équation (2.2). Grâce à cela, il est donc facile d'identifier la partie linéaire des formes AF1 et AF2 comme étant une approximation linéaire de la fonction originale.

Notons $\widehat{f}(x)$ la forme affine AF1 de f sur $\mathbf{X}$. On remarquera que les composantes f_i de la forme affine dépendent aussi de $\mathbf{X}$:

$$\begin{gathered}\widehat{f}(x) = f_0 + \sum_{i=1}^{n} f_i\epsilon_i + f_\pm\epsilon_\pm, \\ \text{avec } \forall i \in \{0, \ldots, n\}, f_i \in \mathbb{R} \ , \ f_\pm \in \mathbb{R}^+, \\ \forall i \in \{1, \ldots, n\}, \ \epsilon_i = [-1, 1] \text{ et } \epsilon_\pm = [-1, 1].\end{gathered}$$

Par définition, la forme affine AF1 est une fonction d'inclusion, donc :

$$\forall x \in \mathbf{X}, \ f(x) \in f_0 + \sum_{i=1}^{n} f_i\epsilon_i + f_\pm\epsilon_\pm.$$

Mais $\forall z \in [-1, 1]^n, \exists x \in \mathbf{X}, z = \mathcal{T}(x)$ avec $\mathcal{T}$ une fonction affine, ainsi, on obtient :

$$\begin{gathered}\forall x \in \mathbf{X}, \ f(x) \in \left(\sum_{i=1}^{n} f_i\mathcal{T}_i(x_i) + f_0 + f_\pm[-1, 1] \right), \\ \forall x \in \mathbf{X}, \ f(x) - \sum_{i=1}^{n} f_i\mathcal{T}_i(x_i) \in [f_0 - f_\pm, f_0 + f_\pm], \\ \text{avec les } \mathcal{T}_i \text{ les composantes de } \mathcal{T}.\end{gathered} \tag{3.1}$$

Ainsi, la proposition suivante en résulte :

Proposition 3.1.1 *Soit $(f_0, \ldots, f_n, f_\pm)$, la reformulation de f sur $\boldsymbol{X}$ en utilisant la forme affine* **AF1**,

si $\forall x \in \boldsymbol{X}, f(x) \leq 0$ *alors* $\forall z \in [-1,1]^n,\ \sum_{i=1}^n f_i z_i \leq f_\pm - f_0$.
et si $\forall x \in \boldsymbol{X}, f(x) = 0$ *alors* $\forall z \in [-1,1]^n$, $\left\{ \begin{array}{l} \sum_{i=1}^n f_i z_i \leq f_\pm - f_0, \\ -\sum_{i=1}^n f_i z_i \leq f_\pm + f_0. \end{array} \right.$

Proposition 3.1.2 *Soit* $(f_0, \ldots, f_n, f_\pm, f_+, f_-)$*, la reformulation de* f *sur* $\boldsymbol{X}$ *en utilisant la forme affine* **AF2**,
si $\forall x \in \boldsymbol{X}, f(x) \leq 0$ *alors* $\forall z \in [-1,1]^n,\ \sum_{i=1}^n f_i z_i \leq f_\pm + f_- - f_0$.
et si $\forall x \in \boldsymbol{X}, f(x) = 0$ *alors* $\forall z \in [-1,1]^n$, $\left\{ \begin{array}{l} \sum_{i=1}^n f_i z_i \leq f_\pm + f_- - f_0, \\ -\sum_{i=1}^n f_i z_i \leq f_\pm + f_+ + f_0. \end{array} \right.$

Démonstration. En considérant la forme affine AF2 à la place de AF1 dans l'équation (3.1), on obtient l'inclusion suivante :

$$\forall x \in \mathbf{X},\ f(x) - \sum_{i=1}^n f_i \mathcal{T}_i(x_i) \in [f_0 - f_\pm - f_-, f_0 + f_\pm + f_+].$$

■

Appliquée à un problème *explicite* d'optimisation globale avec contrainte (P1), une approximation linéaire de chaque équation f, g_i et h_i est obtenue en utilisant AF1 ou AF2. Chaque contrainte d'inégalité est relâchée par une équation linéaire et chaque contrainte d'égalité par deux équations linéaires. Ainsi, un programme linéaire (P2) est automatiquement généré.

$$\left\{ \begin{array}{lll} \min\limits_{x \in \mathbf{X} \subset \mathbb{R}^n} & f(x) & \\ \text{s.t.} & g_k(x) \leq 0\ , & \forall k \in \{1, \ldots, p\}, \\ & h_l(x) = 0\ , & \forall l \in \{1, \ldots, q\}. \end{array} \right. \quad \text{(P1)} \qquad \left\{ \begin{array}{ll} \min\limits_{z \in [-1,1]^n} & c^T z \\ \text{s.t.} & Az \leq b. \end{array} \right. \quad \text{(P2)}$$

Soit $(F_0, \ldots, F_\pm)$, les composantes de la forme affine **AF1** de F tel que $\widehat{F}(x) = F_0 + \sum_{i=1}^n F_i \epsilon_i + F_\pm \epsilon_\pm$, alors, en remplaçant F par f, $g_1, \ldots, g_p$, et $h_1, \ldots, h_q$, le programme linéaire (P2) est construit comme suit :

$$c = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \qquad A = \begin{pmatrix} (g_1)_1 & \ldots & (g_1)_n \\ & \vdots & \\ (g_p)_1 & \ldots & (g_p)_n \\ (h_1)_1 & \ldots & (h_1)_n \\ -(h_1)_1 & \ldots & -(h_1)_n \\ & \vdots & \\ (h_q)_1 & \ldots & (h_q)_n \\ -(h_q)_1 & \ldots & -(h_q)_n \end{pmatrix} \qquad b = \begin{pmatrix} (g_1)_\pm - (g_1)_0 \\ \vdots \\ (g_p)_\pm - (g_p)_0 \\ (h_1)_\pm - (h_1)_0 \\ (h_1)_\pm + (h_1)_0 \\ \vdots \\ (h_q)_\pm - (h_q)_0 \\ (h_q)_\pm + (h_q)_0 \end{pmatrix}$$

Soit $(F_0, \ldots, F_-)$ les composantes de la formes affines **AF2** de F tel que $\widehat{F}(x) = F_0 + \sum_{i=1}^n F_i \epsilon_i + F_\pm \epsilon_\pm + F_+ \epsilon_+ + F_- \epsilon_-$, alors le programme linéaire (P2) est construit comme suit :

$$c = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \quad A = \begin{pmatrix} (g_k)_1 & \dots & (g_k)_n \\ & \vdots & \\ (h_l)_1 & \dots & (h_l)_n \\ -(h_l)_1 & \dots & -(h_l)_n \\ & \vdots & \end{pmatrix} \quad b = \begin{pmatrix} (g_k)_\pm + (g_k)_- - (g_k)_0 \\ \vdots \\ (h_l)_\pm + (h_l)_- - (h_l)_0 \\ (h_l)_\pm + (h_l)_+ + (h_l)_0 \\ \vdots \end{pmatrix}$$

Remarque 3.1.3 *La taille du programme linéaire (P2) reste raisonnable. Le nombre de variables est le même que celui du problème de départ et le nombre de contraintes d'inégalités est au plus le double de celui du problème de départ (P1). Étant donné que les problèmes considérés n'excèdent que rarement 20 variables et 50 contraintes, on peut considérer que les programmes linéaires seront toujours de petite taille, et donc, généralement, plus faciles à résoudre.*

Notons S_1 l'ensemble des solutions réalisables du problème initiale (P1), S_2 l'ensemble des solutions réalisables du programme linéaire (P2), $\mathcal{T}$ la transformation affine de l'espace d'étude $\mathbf{X}$ vers $\varepsilon = [-1, 1]^n$ et E_f la borne inférieure de l'erreur générée par la forme affine de f. En utilisant AF2, on a $E_f = \underline{f_0 + f_\pm \epsilon_\pm} = f_0 - f_\pm$ et avec AF2, $E_f = \underline{f_0 + f_\pm \epsilon_\pm + f_+ \epsilon_+ + f_- \epsilon_-} = f_0 - f_\pm - f_-$.

Proposition 3.1.4 *Supposons qu'il existe x une solution réalisable du problème initial (P1) alors $z = \mathcal{T}(x)$ est une solution réalisable du programme linéaire (P2) et ainsi, on a : $\mathcal{T}(S_1) \subseteq S_2$.*

Démonstration. Supposons que le problème (P1) a été reformulé en utilisant la forme affine AF1. Considérons une contrainte d'inégalité g du problème (P1). Soit x un point vérifiant $g(x) \leq 0$, et z tel que $z = \mathcal{T}(x)$. D'après les propositions 3.1.1, on a : $\sum_{i=1}^{n} g_i z_i \leq g_\pm - g_0$. De même avec un point x vérifiant une contrainte d'égalité h, on a : $\begin{cases} \sum_{i=1}^{n} h_i z_i \leq h_\pm - h_0 \\ -\sum_{i=1}^{n} h_i z_i \leq h_\pm + h_0. \end{cases}$ avec $z = \mathcal{T}(x)$.

Ainsi, soit x une solution réalisable du problème (P1), donc x vérifie chacune des contraintes du problème, ainsi $z = \mathcal{T}(x)$ vérifie chacune des contraintes linéaires générées du programme linéaire (P2). Donc, z est une solution réalisable du programme linéaire (P2).

La démonstration pour la formulation avec la forme affine AF2 suit le même principe. ∎

Corollaire 3.1.5 *Si le programme linéaire (P2) généré à partir d'un problème (P1) n'a pas de solution réalisable sur $\varepsilon = \mathcal{T}(\boldsymbol{X})$, alors le problème (P1) n'a pas de solution réalisable sur $\boldsymbol{X}$.*

Démonstration. En utilisant directement la proposition 3.1.4, on obtient : $S_2 = \emptyset$ implique $S_1 = \emptyset$. ∎

Proposition 3.1.6 *Si z_{sol} est une solution qui minimise le programme linéaire (P2), alors :*

$$\forall x \in S_1,\ f(x) \geq c^T z_{sol} + E_f,$$

$$\text{avec } E_f = f_0 - f_\pm \text{ si l'on utilise AF1,}$$
$$\text{ou } E_f = f_0 - f_\pm - f_- \text{ si l'on utilise AF2.}$$

Démonstration. En utilisant la proposition 3.1.4, on a : $\forall x \in S_1, z = \mathcal{T}(x) \in S_2$. De plus, z_{sol} est une solution qui minimise le programme linéaire (P2), ainsi l'on obtient $\forall z \in S_2, c^T z \geq c^T z_{sol}$. En utilisant la proposition 3.1.1 ou 3.1.2, on a : $\forall x \in S_1, \exists z \in [-1,1], f(x) - c^T z \geq E_f$ et donc $\forall x \in S_1, f(x) \geq c^T z_{sol} + E_f$. ∎

Remarque 3.1.7 *On remarque que, pour la proposition 3.1.6, l'égalité est réalisée lorsque le problème (P1) est linéaire, car dans ce cas, AF1 et AF2 sont simplement une réécriture de (P1) sur $\varepsilon = [-1,1]^n$.*

Proposition 3.1.8 *Considérons un problème (P1) polynomial ; i.e. f et tous les g_i, h_j sont des fonctions polynomiales. Notons z_{AF1} et z_{AF2} les solutions minimales du programme linéaire (P2) généré en utilisant les formes AF1 et respectivement AF2. De plus, en utilisant les notations c_{AF1}, $E_{f_{AF1}}$ et c_{AF2}, $E_{f_{AF2}}$ pour les différentes reformulations de f avec AF1 ou AF2, nous avons :*

$$\forall x \in S_1, f(x) \geq c_{AF2}^T z_{AF2} + E_{f_{AF2}} \geq c_{AF1}^T z_{AF1} + E_{f_{AF1}}.$$

Démonstration. Par construction des arithmétiques AF1 et AF2 définies au chapitre 2 et dans [69], et la proposition 2.2.3, si $z \in S_2$, on a :

$$c_{AF2}^T z + E_{f_{AF2}} \geq c_{AF1}^T z + E_{f_{AF1}},$$
$$c_{AF2}^T z \geq c_{AF2}^T z_{AF2} \text{ et } c_{AF1}^T z \geq c_{AF1}^T z_{AF1}.$$

La proposition 3.1.4 nous donne : $\forall x \in S_1,\ z = \mathcal{T}(X) \in S_2$, et donc :

$$\forall x \in S_1, f(x) \geq c_{AF2}^T z_{AF2} + E_{f_{AF2}} \geq c_{AF1}^T z_{AF2} + E_{f_{AF1}} \geq c_{AF1}^T z_{AF1} + E_{f_{AF1}}.$$

∎

Remarque 3.1.9 *La proposition 3.1.8 peut être généralisée à toutes les fonctions* explicites*, cela dépend simplement de la définition des opérateurs dans les arithmétiques AF1 et AF2. Dans [69], seuls les opérateurs affines et la multiplication étaient pris en compte.*

Proposition 3.1.10 *Si une contrainte du problème (P1) est satisfaite sur $\boldsymbol{X}$ (preuve par analyse d'intervalles), les contraintes linéaires associées peuvent être supprimées du programme linéaire (P2) correspondant.*

Démonstration. Si une contrainte du problème original (P1) est satisfaite sur $\mathbf{X}$ (en le certifiant par analyse d'intervalles), les contraintes linéaires correspondantes sont donc toujours satisfaites pour toutes les valeurs de $z \in [-1,1]^n$ et il n'est pas nécessaire de l'insérer dans la génération du programme linéaire (P2). ∎

Exemple 3.1.11 *Voici un exemple détaillé de l'utilisation de la technique ART. Considérons le problème suivant :*

$$\begin{cases} \min\limits_{x\in \boldsymbol{X}=[1,1.5]\times[4.5,5]\times[3.5,4]\times[1,1.5]} & x_3 + (x_1+x_2+x_3)x_1x_4 \\ s.t. & c_1(x) = x_1x_2x_3x_4 \geq 25, \\ & c_2(x) = x_1^2+x_2^2+x_3^2+x_4^2 = 40, \\ & c_3(x) = 5x_1^4 - 2x_2^3 + 11x_3^2 + 6e^{x_4} \leq 50. \end{cases}$$

Premièrement, on vérifie grâce à l'arithmétique d'intervalles si une contrainte est satisfaite. $C_1(\boldsymbol{X}) = [15.75, 45.0]$, $C_2(\boldsymbol{X}) = [34.5, 45.5]$ *et* $C_3(\boldsymbol{X}) = [-93.94030, 45.95264]$. c_3 *est satisfaite pour tout* $x \in \boldsymbol{X}$*, il n'est donc pas nécessaire de la linéariser.*

En utilisant la forme affine AF1, les reformulations linéaires des équations précédentes donnent :

$$\begin{aligned} x_3 + (x_1+x_2+x_3)x_1x_4 &\longrightarrow 18.98 + 3.44\epsilon_1 + 0.39\epsilon_2 + 0.64\epsilon_3 + 3.05\epsilon_4 + 1.13\epsilon_{\pm}, \\ 25 - x_1x_2x_3x_4 &\longrightarrow -2.83 - 5.57\epsilon_1 - 1.46\epsilon_2 - 1.86\epsilon_3 - 5.57\epsilon_4 + 2.71\epsilon_{\pm}, \\ x_1^2+x_2^2+x_3^2+x_4^2 - 40 &\longrightarrow -0.25 + 0.62\epsilon_1 + 2.38\epsilon_2 + 1.88\epsilon_3 + 0.63\epsilon_4 + 0.25\epsilon_{\pm}. \end{aligned}$$

Il nous est donc possible de générer automatiquement le programme linéaire suivant, qui correspond à une relaxation linéaire du problème précédent :

$$\begin{cases} \min\limits_{z\in[-1,1]^4} & 3.44z_1 + 0.39z_2 + 0.64z_3 + 3.05z_4 & \\ s.t. & -5.57z_1 - 1.46z_2 - 1.86z_3 - 5.57z_4 & \leq 5.54, \\ & 0.62z_1 + 2.38z_2 + 1.88z_3 + 0.63z_4 & \leq 0.5, \\ & -0.62z_1 - 2.38z_2 - 1.88z_3 - 0.63z_4 & \leq 0. \end{cases}$$

Après résolution du programme linéaire, on obtient la solution optimale suivante :

$$z_{sol} = (-1, -0.24, 1, -0.26)\ , \quad c^T z_{sol} = -3.70\ .$$

Ainsi, en utilisant la proposition 3.1.6, on obtient **14.15** *comme minorant de la fonction objectif du problème de départ. En comparant, le minorant obtenu en utilisant directement l'arithmétique d'intervalles, on a* 12.5 *et* 10.34375 *en utilisant l'arithmétique affine AF1 seule.*
On remarque donc que le minorant obtenu avec notre méthodologie est de meilleure qualité que ceux obtenus en utilisant directement l'arithmétique d'intervalles ou AF1. Ceci vient du fait que dans notre approche nous ne considérons pas uniquement la fonction objectif pour calculer le minorant, mais nous incluons aussi les contraintes et l'ensemble des solutions réalisables.

Remarque 3.1.12 *Cette section définit une méthodologie pour construire des relaxations linéaires en utilisant les différentes formes affines et leurs arithmétiques correspondantes. Ce résultat peut être étendu à l'utilisation d'autres formes affines comme celles définies dans le chapitre 2, ou celles des articles [69, 75] qui utilisent une forme quadratique pour le calcul de l'erreur.*

Remarque 3.1.13 *L'expression du programme linéaire généré dépend toujours du domaine sur lequel la reformulation est réalisée. Ainsi, chaque fois que le domaine change, le programme linéaire (P2) doit être regénéré pour approximer au mieux le problème de départ (P1). C'est pourquoi les implémentations des arithmétiques utilisées doivent être performantes.*

3.2 Technique de Relaxation Affine Robuste : $rART_{\text{rAF}}$

La méthodologie expliquée dans la section 3.1 est déjà intéressante en soi : (i) les contraintes sont prises en compte dans le calcul des minorants de la fonction objectif, (ii) la taille des programmes linéaires générés reste raisonnable et (iii) les dépendances linéaires entre les variables sont exploitées dans les reformulations. Mais cette méthode n'est pas entièrement robuste aux erreurs numériques, ceci est notamment dû aux approximations générées par l'utilisation de l'arithmétique flottante. Dans cette section, nous allons expliquer comment rendre notre méthode complètement robuste et fiable.

Avant tout, la première étape consiste à utiliser des arithmétiques robustes, c'est pourquoi nous allons utiliser l'une des arithmétiques affines robustes décrites dans la section 2.5, à la place des arithmétiques standards. Dans un premier temps, nous avons choisi d'utiliser l'arithmétique nommée *reliable Affine Arithmétique* (rAF) décrite dans la section 2.5.2. Ainsi, en utilisant le principe de cette arithmétique, on peut obtenir une version robuste de AF1 et de AF2, que l'on nomme *rAF1* et *rAF2*. De plus, tout comme dans la section 3.1, les propositions 3.1.1 et 3.1.2 peuvent être réécrites en utilisant *rAF1* et *rAF2*. Ceci permet d'obtenir une reformulation affine robuste de n'importe laquelle *fonction explicite*, c'est-à-dire que l'on obtient une relaxation linéaire dans laquelle toutes les constantes sont des intervalles. En conséquence, en utilisant la méthodologie de reformulation décrite précédemment avec *rAF1* et *rAF2*, on peut générer automatiquement un programme linéaire robuste, c'est-à-dire un programme linéaire (P2) dans lequel toutes les constantes sont des intervalles et avec lequel la proposition 3.1.4 est vérifiée de manière fiable.

Lorsque le programme linéaire robuste est généré, deux approches sont possibles pour le résoudre.

– La première consiste à utiliser un solveur linéaire par intervalles tel que LURUPA [49, 55, 56] pour obtenir un minorant fiable de la fonction objectif ou un certificat d'infaisabilité. Puis, ces propriétés sont étendues au problème général (P1) en utilisant la proposition 3.1.6 et le corollaire 3.1.5.

– Pour la seconde approche, on génère aussi une reformulation linéaire fiable sur le même principe que précédemment. Puis, on utilise les formules de conversion rAF1/AF1 ou rAF2/AF2 (voir équation (2.1)) pour obtenir une reformulation linéaire dans laquelle toutes les variables sont des scalaires, mais dans cette reformulation, toutes les erreurs numériques ont été prises en compte par l'arithmétique affine robuste et ont été transférées dans le dernier terme de la forme affine (correspondant à l'erreur générée). Ainsi, nous avons un programme linéaire classique qui vérifie de manière robuste et fiable la proposition 3.1.4. Puis, nous utilisons les résultats de Neumaier et Shcherbina [82] pour calculer un minorant fiable de notre programme linéaire ou un certificat d'infaisabilité.

Remarque 3.2.1 *Dans la deuxième approche, il est possible de générer le programme linéaire directement en utilisant les autres arithmétiques affines robustes, telles que la* self-validated affine arithmetic *et la* floating-point affine arithmetic. *Le programme généré serait alors complètement fiable et robuste, et les travaux de Neumaier et Shcherbina pourraient s'appliquer également.*

Notons S_1 l'ensemble des solutions réalisables du problème initial (P1), S_2 l'ensemble des solutions réalisables du programme linéaire (P2).

Les travaux de Neumaier et Shcherbina appliqués à notre cas particulier peuvent être résumés de la manière suivante :

$$\left\{ \begin{array}{cc} \min\limits_{z\in\varepsilon=[-1,1]^n} & c^T z \\ \text{s. t.} & Az \leq b. \end{array} \right. \quad \text{(P2)} \qquad \left\{ \begin{array}{cc} \min\limits_{\lambda\in\mathbb{R}^m_+, u, l\in\mathbb{R}^n_+} & b^T\lambda + \sum\limits_{i=1}^{n}(l_i+u_i) \\ \text{s. t.} & A^T\lambda - l + u = -c. \end{array} \right. \quad \text{(D2)}$$

Le programme linéaire (D2) correspond à la formulation duale du programme linéaire (P2). Soit (λ_S, l_S, u_S) une solution approchée du dual donnée par un solveur linéaire, et $(\Lambda_S, \mathbf{L_S}, \mathbf{U_S})$ l'extension de cette solution en arithmétique d'intervalles ; cette conversion consiste à remplacer les nombres flottants par un intervalle constitué de deux fois le même nombre, ceci a pour but de forcer tous les calculs en arithmétique d'intervalles. Il est donc possible de calculer le résidu r du problème dual (D2) par arithmétique d'intervalles, tel que :

$$r \in \mathbf{R} = c + A^T\Lambda_S - \mathbf{L_S} + \mathbf{U_S}. \tag{3.2}$$

Ainsi, en utilisant la borne calculée dans [82], on obtient :

$$\begin{array}{c} \forall z \in S_2,\ c^T z \in \left(\mathbf{R}^T\varepsilon - {\Lambda_S}^T[-\infty, b] + {\mathbf{L_S}}^T\varepsilon - {\mathbf{U_S}}^T\varepsilon\right), \\ \text{avec } \varepsilon = ([-1,1],\ldots,[-1,1])^T \text{ et } [-\infty, b] = ([-\infty, b_1],\ldots,[-\infty, b_m])^T. \end{array} \tag{3.3}$$

Proposition 3.2.2 *Soit $(\Lambda_S, \mathbf{L_S}, \mathbf{U_S})$ l'extension à l'arithmétique d'intervalles de la solution approchée qui minimise le dual du programme linéaire (P2), alors :*

$$\forall x \in S_1\ ,\ f(x) \geq \underline{\left(\mathbf{R}^T\varepsilon - {\Lambda_S}^T[-\infty, b] + {\mathbf{L_S}}^T\varepsilon - {\mathbf{U_S}}^T\varepsilon\right) + E_f}.$$

Démonstration. Ce résultat s'obtient par application des équations (3.3) pour obtenir le minorant de la fonction objectif du programme linéaire (P2) et de la proposition 3.1.6 pour ajouter l'erreur de la reformulation et obtenir un minorant de la fonction objectif du problème original (P1). ∎

Lorsque le minorant ne peut être calculé, le problème dual (D2) est donc soit non borné soit infaisable. Cependant, l'ensemble des solutions réalisables du problème primal (P2) est inclue dans $\varepsilon = [-1, 1]^n$. Donc, le dual (D2) ne peut être infaisable. Ainsi, pour prouver que le dual (D2) est non borné, nous recherchons une solution réalisable du programme linéaire (D3) (c'est une technique bien connue qui est directement adaptée de [82]) :

$$\begin{cases} b^T\lambda + \sum_{i=1}^{n} (l_i + u_i) \neq 0, \\ A^T\lambda - l + u = 0, \\ \lambda \in \mathbb{R}^m_+, u, l \in \mathbb{R}^n_+. \end{cases} \tag{D3}$$

Proposition 3.2.3 *Soit $(\Lambda_c, \mathbf{L_c}, \mathbf{U_c})$ l'extension à l'arithmétique d'intervalles d'une solution réalisable du programme linéaire (D3),*
si $0 \notin \left(\left(A^T\Lambda_c - \mathbf{L_c} + \mathbf{U_c}\right)^T\varepsilon - {\Lambda_c}^T[-\infty, b] + {\mathbf{L_c}}^T\varepsilon - {\mathbf{U_c}}^T\varepsilon\right)$, alors le problème de départ (P1) est certifié être irréalisable, c'est-à-dire sans solution réalisable.

Démonstration.
Par application du précédent calcul au résidu du dual $r \in \mathbf{R} = A^T\Lambda_c - \mathbf{L_c} + \mathbf{U_c}$, on obtient que si $0 \notin \left(\left(A^T\Lambda_c - \mathbf{L_c} + \mathbf{U_c}\right)^T\varepsilon - {\Lambda_c}^T[-\infty, b] + {\mathbf{L_c}}^T\varepsilon - {\mathbf{U_c}}^T\varepsilon\right)$, le programme primal (P2) est certifié être irréalisable. Par application du corollaire 3.1.5, la proposition 3.2.3 est prouvée. ∎

Ainsi en utilisant les propositions 3.2.2 et 3.2.3, nous avons une manière fiable et robuste de calculer des certificats d'infaisabilité et des minorants de la fonction objectif en prenant en compte les contraintes.

3.3 Technique de Reformulation Affine Mixte : *MART*

Cette technique est une extension de la technique ART, décrite dans la section 3.1, pour considérer des problèmes d'optimisation globale mixte avec des variables entières et continues, tels que le problème (MINLP). Elle permet de générer une relaxation linéaire mixte.

$$\begin{cases} \min\limits_{(x,y)\in\mathbf{X}\times\mathbf{Y}\subset\mathbb{R}^n\times\mathbb{Z}^m} & f(x,y) & \\ \text{s.t.} & g_i(x,y) \leq 0\ , & \forall i \in \{1,\ldots,p\}, \\ & h_j(x,y) = 0\ , & \forall j \in \{1,\ldots,q\}. \end{cases} \tag{MINLP}$$

Tout comme dans la technique ART, les extensions de l'arithmétique affine fixent le nombre de variables ϵ_i égal à la taille du problème (MINLP). Ainsi, on obtient une transformation affine $\mathcal{T}$ entre un domaine de départ dans $\mathbf{X} \times \mathbf{Y} \subset \mathbb{R}^n \times \mathbb{Z}^m$ et $\epsilon = [-1, 1]^{n+m}$. Cette transformation affine $\mathcal{T}$ correspond à la conversion d'un intervalle en forme affine. La technique de reformulation affine mixte procède ainsi :

- La technique ART génère une relaxation linéaire du problème original sur le domaine $\mathbf{X} \times \mathbf{Y} \subset \mathbb{R}^n \times \mathbb{Z}^m$ que l'on relaxe dans $\mathbb{R}^{n+m}$. L'espace d'étude du programme ainsi généré est $[-1, 1]^{n+m}$.
- La transformation $\mathcal{T}^{-1}$ est utilisée pour retransformer les variables entières vers leur domaine original. Ainsi, le domaine du programme linéaire généré est $[-1, 1]^n \times \mathbf{Y}$.
- Puis, dans le programme linéaire généré, il suffit de considérer comme entières les variables entières du problème original. On obtient alors un programme linéaire mixte.

Dans la proposition 3.3.2, nous reformulons le procédé de génération des relaxations linéaires de la proposition 3.1.2. La transformation $\mathcal{T}^{-1}$ est seulement utilisée sur les variables entières.

Remarque 3.3.1 *Afin d'éviter les redondances, nous ne considérerons que les formes affines AF2. Bien entendu, toutes les propositions suivantes peuvent se réécrire en utilisant les autres extensions de l'arithmétique affine, notamment AF1, AF3, IA_AF1, IA_AF2, etc.*

Rappelons que $w(\mathbf{x})$ représente la largeur de l'intervalle $\mathbf{x}$ et $mid(\mathbf{x})$ son milieu. Notons $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \subset \mathbb{R}^n$ et $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m) \subset \mathbf{Z}^m$ les composantes des domaines d'études.

Proposition 3.3.2 *Considérons $(f_0, \ldots, f_{n+m}, f_\pm, f_+, f_-)$ la reformulation de f sur $\boldsymbol{X} \times \boldsymbol{Y} \subset \mathbb{R}^n \times \mathbb{Z}^m$ en utilisant* **AF2**,
ainsi, si $\forall (x, y) \in \boldsymbol{X} \times \boldsymbol{Y}$, $f(x, y) \leq 0$, alors $\forall (z, y) \in [-1, 1]^n \times \boldsymbol{Y}$,

$$\sum_{i=1}^{n} f_i z_i + \sum_{i=n+1}^{n+m} \frac{2f_i}{w(\mathbf{y}_i)} y_i \leq \sum_{i=n+1}^{n+m} \frac{2f_i mid(\mathbf{y}_i)}{w(\mathbf{y}_i)} + f_\pm + f_- - f_0 \ ,$$

et si $\forall (x, y) \in \boldsymbol{X} \times \boldsymbol{Y}, f(x, y) = 0$, alors $\forall (z, y) \in [-1, 1]^n \times \boldsymbol{Y}$,

$$\begin{cases} \displaystyle\sum_{i=1}^{n} f_i z_i + \sum_{i=n+1}^{n+m} \frac{2f_i}{w(\mathbf{y}_i)} y_i \leq \sum_{i=n+1}^{n+m} \frac{2f_i mid(\mathbf{y}_i)}{w(\mathbf{y}_i)} + f_\pm + f_- - f_0 \ , \\ \displaystyle -\sum_{i=1}^{n} f_i z_i - \sum_{i=n+1}^{n+m} \frac{2f_i}{w(\mathbf{y}_i)} y_i \leq \sum_{i=n+1}^{n+m} \frac{-2f_i mid(\mathbf{y}_i)}{w(\mathbf{y}_i)} + f_\pm + f_+ + f_0. \end{cases}$$

Démonstration. Notons $\widehat{f}(x, y)$ la forme affine AF2 de f sur $\mathbf{X} \times \mathbf{Y} \subset \mathbb{R}^n \times \mathbb{Z}^m$. Remarquons que les composantes f_i dépendent aussi de $\mathbf{X} \times \mathbf{Y}$. Pour $i \in \{1, \ldots, n\}$,

les variables ϵ_i sont associées aux intervalles $\mathbf{x}_i \subset \mathbb{R}$ et pour $i \in \{1,\ldots,m\}$, les ϵ_{n+i} aux $\mathbf{y}_i \subset \mathbb{Z}$.

$$\widehat{f}(x,y) = f_0 + \sum_{i=1}^{n} f_i\epsilon_i + \sum_{i=n+1}^{n+m} f_i\epsilon_i + f_\pm\epsilon_\pm + f_+\epsilon_+ + f_-\epsilon_-,$$
$$\text{avec } \forall i \in \{0,\ldots,n+m\}, f_i \in \mathbb{R}\ ,\ f_\pm, f_+, f_- \in \mathbb{R}^+,$$
$$\forall i \in \{1,\ldots,n+m\},\ \epsilon_i = [-1,1] \text{ et } \epsilon_\pm = [-1,1], \epsilon_+ = [0,1], \epsilon_- = [-1,0].$$

Par définition, la forme affine AF2 est une fonction d'inclusion :

$$\forall (x,y) \in \mathbf{X}\times\mathbf{Y},\ f(x,y) \in f_0 + \sum_{i=1}^{n+m} f_i\epsilon_i + f_\pm\epsilon_\pm + f_+\epsilon_+ + f_-\epsilon_-.$$

Mais $\forall z \in [-1,1]^n, \exists x \in \mathbf{X}, \forall i \in \{1,\ldots,n\}, z_i = \mathcal{T}_i(x_i) = \dfrac{2}{w(\mathbf{x}_i)}(x_i - mid(\mathbf{x}_i))$.
De même, il existe une transformation affine entre $\mathbf{Y}$ et $[-1,1]^m$, $\forall (x,y) \in \mathbf{X}\times\mathbf{Y}$, nous avons :

$$f(x,y) \in \left(\sum_{i=1}^{n} f_i\mathcal{T}_i(x_i) + \sum_{i=n+1}^{n+m} f_i\mathcal{T}_i(y_i) + f_0 + f_\pm[-1,1] + f_+[0,1] + f_-[-1,0]\right),$$
$$f(x,y) - \sum_{i=1}^{n} f_i\mathcal{T}_i(x_i) - \sum_{i=n+1}^{n+m} f_i\mathcal{T}_i(y_i) \in [f_0 - f_\pm - f_-, f_0 + f_\pm + f_+],$$

$$f(x,y) - \sum_{i=1}^{n} f_i\mathcal{T}_i(x_i) - \sum_{i=n+1}^{n+m} \frac{2f_i}{w(\mathbf{y}_i)}y_i + \sum_{i=n+1}^{n+m} \frac{2f_i mid(\mathbf{y}_i)}{w(\mathbf{y}_i)}$$
$$\in [f_0 - f_\pm - f_-, f_0 + f_\pm + f_+].$$

La proposition 3.3.2 découle directement de l'application de la dernière équation. ∎

Exemple 3.3.3 *Considérons l'équation* $\forall (x,y) \in [0,1]\times(\mathbb{Z}\cap[0,4]), x\times y = 0$. *La première étape consiste à reformuler* $[0,1]$ *et* $[0,4]$ *en forme affine AF2. Puis, la multiplication entre les deux formes affines est calculée et finalement, en utilisant la proposition 3.3.2, les deux inégalités sont générées :*

$$\mathbf{x} = [0,1] \to \widehat{x} = 0.5 + 0.5\epsilon_1 \quad et \quad \mathbf{y} = [0,4] \to \widehat{y} = 2 + 2\epsilon_2,$$
$$\widehat{x}\times\widehat{y} = 1 + \epsilon_1 + \epsilon_2 + \epsilon_\pm,$$
$$\Rightarrow \forall (z,y) \in [-1,1]\times(\mathbb{Z}\cap[0,4]), \begin{cases} z + 0.5y \le 1, \\ -z - 0.5y \le 1. \end{cases}$$

On peut remarquer que les ensembles $\{(z,0), z \in [-1,1]\}$ *et* $\{(-1,y), y \in \mathbb{Z}\cap[0,4]\}$ *sont des solutions du système d'inéquations, ce qui implique que les ensembles* $\{(x,0), x \in [0,1]\}$ *et* $\{(0,y), y \in \mathbb{Z}\cap[0,4]\}$ *sont des solutions du système de départ.*

Considérons un problème (M1) de type (MINLP). En utilisant la proposition 3.3.2, on peut construire un programme linéaire mixte à variables continues et entières (M2), qui est une relaxation linéaire de (M1) :

$$\left\{\begin{array}{ll} \min\limits_{(x,y)\in \mathbf{X}\times\mathbf{Y}\subset\mathbb{R}^n\times\mathbb{Z}^m} & f(x,y) \\ \text{s.t.} & \\ \forall k\in\{1,\dots,p\}, & g_k(x,y)\leq 0, \\ \forall l\in\{1,\dots,q\}, & h_l(x,y)=0. \end{array}\right. \quad \text{(M1)} \qquad \left\{\begin{array}{ll} \min\limits_{(z,y)\in[-1,1]^n\times\mathbf{Y}} & c^Tz+d^Ty \\ \text{s.t.} & A\begin{pmatrix} z \\ y \end{pmatrix}\leq b. \end{array}\right. \quad \text{(M2)}$$

Notons $(F_0,\dots,F_-)$, le résultat de la forme affine **AF2** de F tel que $\widehat{F}(x) = F_0+\sum\limits_{i=1}^{n+m}F_i\epsilon_i+F_\pm\epsilon_\pm+F_+\epsilon_++F_-\epsilon_-$, ainsi le programme linéaire (M2) est construit comme suit :

$$c=(f_1,\dots,f_n) \qquad d=\left(\frac{2f_{n+1}}{w(\mathbf{y}_1)},\dots,\frac{2f_{n+m}}{w(\mathbf{y}_n)}\right)$$

$$A=\begin{pmatrix} (g_k)_1 & \dots & (g_k)_n & \frac{2(g_k)_{n+1}}{w(\mathbf{y}_1)} & \dots & \frac{2(g_k)_{n+m}}{w(\mathbf{y}_n)} \\ \vdots & & \vdots & \vdots & & \vdots \\ (h_l)_1 & \dots & (h_l)_n & \frac{2(h_l)_{n+1}}{w(\mathbf{y}_1)} & \dots & \frac{2(h_l)_{n+m}}{w(\mathbf{y}_n)} \\ -(h_l)_1 & \dots & -(h_l)_n & \frac{-2(h_l)_{n+1}}{w(\mathbf{y}_1)} & \dots & \frac{-2(h_l)_{n+m}}{w(\mathbf{y}_n)} \\ \vdots & & \vdots & \vdots & & \vdots \end{pmatrix}$$

$$b=\begin{pmatrix} \sum_{i=1}^m \frac{2(g_k)_{n+i}mid(\mathbf{y}_i)}{w(\mathbf{y}_i)}+(g_k)_\pm+(g_k)_--(g_k)_0 \\ \vdots \\ \sum_{i=1}^m \frac{2(h_l)_{n+i}mid(\mathbf{y}_i)}{w(\mathbf{y}_i)}+(h_l)_\pm+(h_l)_--(h_l)_0 \\ -\sum_{i=1}^m \frac{2(h_l)_{n+i}mid(\mathbf{y}_i)}{w(\mathbf{y}_i)}+(h_l)_\pm+(h_l)_++(h_l)_0 \\ \vdots \end{pmatrix}$$

En résolvant le programme (M2), on peut en déduire que si (M2) n'a pas de solution réalisable, le problème (M1) n'en a pas non plus. Si une solution réalisable de (M2) existe, un minorant de la fonction objectif de (M1) peut être calculé en utilisant la proposition suivante :

Proposition 3.3.4 *Soit (z_{sol},y_{sol}) une solution réalisable qui minimise le programme linéaire mixte (M2) et S l'ensemble des solutions réalisables du problème de départ (M1),*

$$\forall x\in S\ ,\ f(x)\geq c^Tz_{sol}+d^Ty_{sol}+f_0-f_\pm-f_--\sum_{i=1}^m\frac{2f_{n+i}mid(\mathbf{y}_i)}{w(\mathbf{y}_i)}.$$

Tout comme ART, cette nouvelle technique de relaxation MART génère des programmes linéaires de petite taille. Par la suite, nous l'avons incluse dans notre

algorithme de Branch and Bound par intervalles *IBBA*, afin de tester son efficacité. L'idée principale de cette approche est de générer de petits programmes linéaires mixtes sur chaque sous-domaine, qui pourront être résolus assez rapidement afin d'améliorer le calcul des minorants et d'éliminer des éléments ne contenant pas le minimum global. Ainsi, cette technique doit être utilisée dans les cas où il est préférable de subdiviser le domaine plutôt que d'ajouter d'autres plans de coupe pour améliorer la relaxation linéaire, comme cela peut se faire dans les techniques basées sur une approche RLT standard.

Remarque 3.3.5 *La technique robuste rART et la technique mixte MART peuvent bien entendu être combinées et nommées rMART. Il est donc possible d'obtenir des relaxations linéaires mixtes robustes. Pour cela, il suffit d'utiliser rAF2 à la place de AF2, puis d'adapter les propositions 3.2.2 et 3.3.4 en les combinant.*

3.4 Intégration dans un algorithme de Branch and Bound par intervalles

Afin de prouver l'efficacité de nos méthodes de reformulation décrites précédemment, nous les avons insérées dans notre algorithme de Branch and Bound par intervalles, nommé *IBBA* dont le principe est décrit dans la section 1.2. Notons tout de même qu'il existe de nombreux autres algorithmes de Branch and Bound basés sur l'arithmétique d'intervalles, par exemple *GlobSol* développé par Kearfott et al. [52], dans lesquelles nos méthodes peuvent très facilement être intégrées. Le principe fondamental reste le même, seules les techniques d'accélération diffèrent.

L'algorithme *IBBA* est résumé dans l'algorithme 2 du chapitre 1. L'un des principaux avantages de cet algorithme réside dans sa modularité. En effet, il est très facile d'insérer ou de supprimer des techniques d'accélération. Par exemple à la ligne 6 de l'algorithme 2, la technique de propagation des contraintes est incluse pour réduire la taille des éléments traités, mais par la suite, certains tests numériques ont été réalisés en supprimant cette technique.

La technique de reformulation affine ART décrite dans les sections précédentes peut être aussi facilement insérée dans cet algorithme 2. Elle intervient au niveau des lignes 7 et 8 de l'algorithme 2. A chaque itération, pour chaque élément V_1 et V_2, le programme linéaire associé (P2) est automatiquement généré et un solveur linéaire (tel que C-PLEX) le résout. Si le programme linéaire est sans solution réalisable, l'élément est éliminé. Dans le cas contraire, la solution du programme linéaire est utilisée pour calculer un minorant de la fonction objectif sur les sous-domaines $\mathbf{Z}_1$ ou $\mathbf{Z}_2$.

Remarque 3.4.1 *Afin de prendre en compte la valeur du minimum courant dans la technique de reformulation affine, l'équation $f(x) \leq \tilde{f}$ est ajoutée à l'ensemble des contraintes lorsque $\tilde{f} \neq +\infty$.*

Algorithme 8 Technique de Reformulation Affine basée sur l'Arithmétique Affine : ART_{AF}

1: Soit $V = (\mathbf{Z}, f_z, R)$ l'élément courant avec f_z un minorant de f sur $\mathbf{Z}$,
2: Initialiser un programme linéaire avec le même nombre de variables que le problème de départ (P1),
3: Générer la forme affine de f en utilisant AF1 ou AF2,
4: Expliciter c la fonction objectif de (P2) et E_f la borne inférieure de l'erreur de la forme affine de f, $c = (f_1, \ldots, f_n)$, $E_f = f_0 - f_\pm$ avec AF1 et $E_f = f_0 - f_\pm - f_-$ avec AF2,
5: **pour** toute contrainte g du problème du départ (P1) **faire**
6: Calcul de G($\mathbf{Z}$) l'extension naturelle aux intervalles de g sur $\mathbf{Z}$,
7: **si** g n'est pas satisfaite par intervalles **alors**
8: Générer la forme affine de g en utilisant AF1 ou AF2,
9: Ajouter les contraintes dans le programme linéaire (P2), comme décrit dans la section 3.1,
10: **finsi**
11: **fin pour**
12: Résoudre le programme linéaire (P2) ainsi généré avec un solveur linéaire tel que C-PLEX,
13: **si** le programme linéaire a une solution z_{sol} **alors**
14: $f_z = max(f_z, c^T z_{sol} + E_f)$,
15: **sinon si** le programme linéaire est irréalisable **alors**
16: Éliminer l'élément $V = (\mathbf{Z}, f_z, R)$
17: **finsi**

L'algorithme 8 décrit toutes les étapes de la technique de reformulation affine ART_{AF}. Le but de cette méthode est d'accélérer la résolution du problème en réduisant le nombre d'itérations et le temps de calcul de *IBBA* (algorithme 2). A la ligne 7 de l'algorithme 8, la proposition 3.1.10 est utilisée pour réduire le nombre de contraintes ; ceci permet de limiter la taille du programme linéaire sans perdre d'information. Le calcul réalisé en ligne 14 fournit un minorant de la fonction objectif du problème général sur le sous-domaine $\mathbf{Z}$ et ceci est prouvé grâce à la proposition 3.1.6. L'élimination en ligne 16 est déduite du corollaire 3.1.5. Si le solveur linéaire ne trouve pas de solution en un temps raisonnable (environ une seconde) ou un nombre d'itérations fixé (pas plus de 10 fois le nombre de variables), le minorant f_z n'est pas modifié et l'algorithme 2 continue comme si, à cette itération uniquement, la technique n'était pas activée.

Remarque 3.4.2 *L'arithmétique affine n'a pas de définition pour les fonctions allant à l'infini ; par exemple, lorsqu'une division par zéro apparaît. Dans ce cas, il est impossible de construire une linéarisation de la fonction. C'est pourquoi, si la fonction objectif a des valeurs non-bornée sur le domaine considéré, le minorant n'est pas modifié et l'algorithme 2 continue sans utiliser la technique de reformulation affine à cette itération. En généralisant, s'il est impossible de linéariser une contrainte,*

la méthode continue sans inclure celle-ci dans le programme linéaire. Ainsi, le programme linéaire est plus relâché et le calcul du minorant de la fonction objectif reste valable.

Afin d'améliorer encore les performances de l'algorithme *IBBA*, nous avons exploité davantage le point z_{sol} obtenu par la résolution du programme linéaire (P2). En effet, ce point vérifie le système généré par les relaxations linéaires des contraintes. Il peut donc être intéressant de tester le point $\mathcal{T}^{-1}(z_{sol})$ dans le problème de départ (P1) pour vérifier s'il est réalisable. Ce test additionnel a été intégré uniquement dans les versions $IBBA{+}ART_{\text{AF2\_primal}}{+}CP$ et $IBBA{+}ART_{\text{AF2\_dual}}{+}CP$ de notre algorithme qui seront testées dans la section 3.5.4. Ce test supplémentaire s'est avéré efficace puisque le nombre d'améliorations du minimum courant par ce test fut équivalent à celui avec le test classique du point du milieu.

Dans la section 3.2, nous avons expliqué comment rendre robuste et fiable la technique de reformulation affine. L'algorithme 9 résume cette méthode que nous avons appelée $rART_{\text{rAF}}$. Il correspond à une adaptation de l'algorithme 8.

Dans un premier temps, nous utilisons rAF1 ou rAF2 avec les conversions entre rAF1/AF1 ou rAF2/AF2 pour produire un programme linéaire (P2), en utilisant l'équation (2.1) et les propositions 3.1.1 et 3.1.2. Puis la proposition 3.1.10 est utilisée ligne 7 de l'algorithme 9 pour réduire le nombre de contraintes, tout comme dans l'algorithme 8. Ainsi, dans la plupart des cas, le nombre de contraintes ajoutées reste faible, et la résolution du dual est donc plus simple. De plus, nous n'avons pas besoin d'expliciter la solution primale, il est donc conseillé de générer directement le problème dual (D2) et de le résoudre avec une résolution primale. Si une solution duale est trouvée, la proposition 3.2.2 certifie que le calcul de la ligne 16 de l'algorithme 9 produit un minorant de la fonction objectif du problème (P1) sur l'intervalle $\mathbf{Z}$. Dans le cas contraire, lorsque le solveur montre que le dual est non borné, la proposition 3.2.3 produit un certificat d'infaisabilité du problème (P1) sur $\mathbf{Z}$ à la ligne 18-24 ; l'élément V pourra donc être supprimé.

Ainsi, dans cette section, nous avons décrit deux nouvelles méthodes d'accélération qui peuvent être ajoutées dans tout algorithme de Branch and Bound par intervalles. $rART_{\text{rAF}}$ (algorithme 9) inclue dans *IBBA* (algorithme 2) permet de garder la propriété de robustesse aux erreurs numériques propre à ce type d'algorithme. Dans la prochaine section, nous étudierons son intégration et son impact sur les temps de résolution et le nombre d'itérations de notre algorithme de Branch and Bound par intervalles.

Algorithme 9 Technique de Reformulation Affine robuste basée sur l'Arithmétique Affine robuste : $rARTR_{\text{rAF}}$

1: Soit $V = (\mathbf{Z}, f_z, R)$ l'élément courant avec f_z un minorant de f sur $\mathbf{Z}$,
2: Initialiser un programme linéaire avec le même nombre de variables que le problème de départ (P1),
3: Générer la forme affine de f en utilisant rAF1 ou rAF2,
4: En utilisant les conversions rAF1/AF1 ou rAF2/AF2, générer E_f et c du programme linéaire (P2),
5: **pour** toute contrainte g du problème générale (P1) **faire**
6: Calculer G($\mathbf{Z}$) l'extension naturelle aux intervalles de g sur $\mathbf{Z}$,
7: **si** g n'est pas satisfaite par intervalles **alors**
8: Générer la forme affine de g en utilisant les formes affines rAF1 ou rAF2 les conversions rAF1/AF1 ou rAF2/AF2,
9: Ajouter les contraintes linéaires associées à cette linéarisation dans le programme linéaire (P2) tel que décrit dans la section 3.1,
10: **finsi**
11: **fin pour**
12: Générer le programme dual (D2) associé au programme linéaire (P2),
13: Résoudre le dual (D2) avec un solveur linéaire primal,
14: **si** le programme dual a une solution (λ_s, l_s, u_s) **alors**
15: $(\Lambda_s, \mathbf{L_s}, \mathbf{U_s})$ = extension de (λ_s, l_s, u_s) en intervalles,
16: $f_z = \max\Big(f_z, \underline{\big(\mathbf{R}^T\varepsilon - {\Lambda_S}^T[-\infty, b] + {\mathbf{L_S}}^T\varepsilon - {\mathbf{U_S}}^T\varepsilon\big) + E_f}\Big)$,
17: **sinon si** le programme dual est irréalisable **alors**
18: résoudre le programme (D3) associé au programme dual (D2) généré précédemment,
19: **si** le programme (D3) a une solution (λ_c, l_c, u_c) **alors**
20: $(\Lambda_c, \mathbf{L_c}, \mathbf{U_c})$ = extension de (λ_c, l_c, u_c) en intervalles,
21: **si** $0 \notin \Big(\big(A^T\Lambda_c - \mathbf{L_c} + \mathbf{U_c}\big)^T\varepsilon - {\Lambda_c}^T[-\infty, b] + {\mathbf{L_c}}^T\varepsilon - {\mathbf{U_c}}^T\varepsilon\Big)$ **alors**
22: Eliminer l'élément $V = (\mathbf{Z}, f_z, R)$
23: **finsi**
24: **finsi**
25: **finsi**

3.5 Tests numériques

Dans cette section, nous considérons 74 problèmes d'optimisation globale non linéaires et non convexes avec contraintes. Ces tests sont extraits de la librairie 1 du site internet COCONUT [81, 83]. Nous nous sommes intéressés essentiellement aux problèmes avec contraintes, ayant moins de 25 variables et sans fonction *cos* et *sin* qui ne sont pas encore implémentées dans nos arithmétiques affines (voir section 2) ; néanmoins, la racine carrée, l'inverse, le logarithme et l'exponentielle sont disponibles. Pour chacun des 74 problèmes tests, les expressions de toutes les équations sont directement recopiées telles qu'elles sont écrites dans le format AMPL du site COCONUT. Aucune modification n'a été faite sur les expressions des fonctions et contraintes, même si pour certains problèmes la formulation est clairement non adaptée au calcul d'encadrement par arithmétique d'intervalles ou affine.

Le code est écrit en Fortran 90/95 et utilise le compilateur *f90* de SUN, avec la librairie incluse permettant d'utiliser l'arithmétique d'intervalles. Pour résoudre le programme linéaire généré par notre technique, on utilise C-PLEX version 11.0. Tous les tests sont réalisés sur un PC-Intel-Xeon-3Gz avec 2 Go de RAM et un système Linux 64 bits. Le test d'arrêt de l'algorithme 2 est sur l'encadrement de la valeur du minimum global : $\left(\tilde{f} - \min_{(\mathbf{Z}, f_z, R) \in \mathcal{L}} f_z \leq \epsilon_f \max(|\tilde{f}|, 1)\right)$. La précision relative demandée sur la fonction objectif est fixée à $\epsilon_f := 10^{-8}$ pour tous les problèmes, et la même valeur est prise pour la satisfaction des contraintes. La précision pour résoudre le programme linéaire avec C-PLEX est fixée à 10^{-8} et nous avons limité le nombre d'itérations du simplexe à 100 fois le nombre de variables et de contraintes. De plus, deux autres limites sont imposées : (a) sur le temps de calcul de *IBBA* (algorithme 2) qui doit être inférieur à 60 minutes et (b) sur le nombre maximal d'éléments pouvant être stockés dans la structure de données $\mathcal{L}$ que nous limitons à 2 millions d'éléments (cela correspond approximativement à la limite en mémoire vive disponible sur notre ordinateur pour les plus gros problèmes). Lorsque le code termine normalement, les valeurs correspondant (i) au fait que le problème soit résolu ou pas, (ii) au nombre d'itérations de la boucle principale de *IBBA* (algorithme 2), et (iii) au temps de calcul en secondes, sont respectivement données dans les colonnes 'ok ?', 'Iter' et 't' des tableaux 3.1, 3.2, 3.3, 3.4 et 3.6. Ces tests de terminaisons sont comparables à ceux proposés dans [53] pour *GlobSol*.

Le nom des problèmes de COCONUT est inscrit dans la première colonne des tableaux ; sur le site COCONUT, tous les problèmes ainsi que les meilleures solutions connues sont donnés. Les colonnes N et M représentent le nombre de variables et le nombre de contraintes pour chaque problème. Le problème test *hs071* vient de la librairie 2 de COCONUT, il correspond à l'exemple 3.1.11 sous les contraintes c_1 et c_2 uniquement sur le domaine $\mathbf{X} = [1, 5]^4$.

Pour chaque tableau et profil de performance, *IBBA+CP* désigne les résultats obtenus pour tous les problèmes en utilisant *IBBA* (algorithme 2) et la technique d'accélération de propagation de contraintes (CP) décrite dans [70] et la section 1.3.2. $IBBA{+}rART_{\text{rAF2}}$ désigne les résultats obtenus avec *IBBA* (algorithme 2)

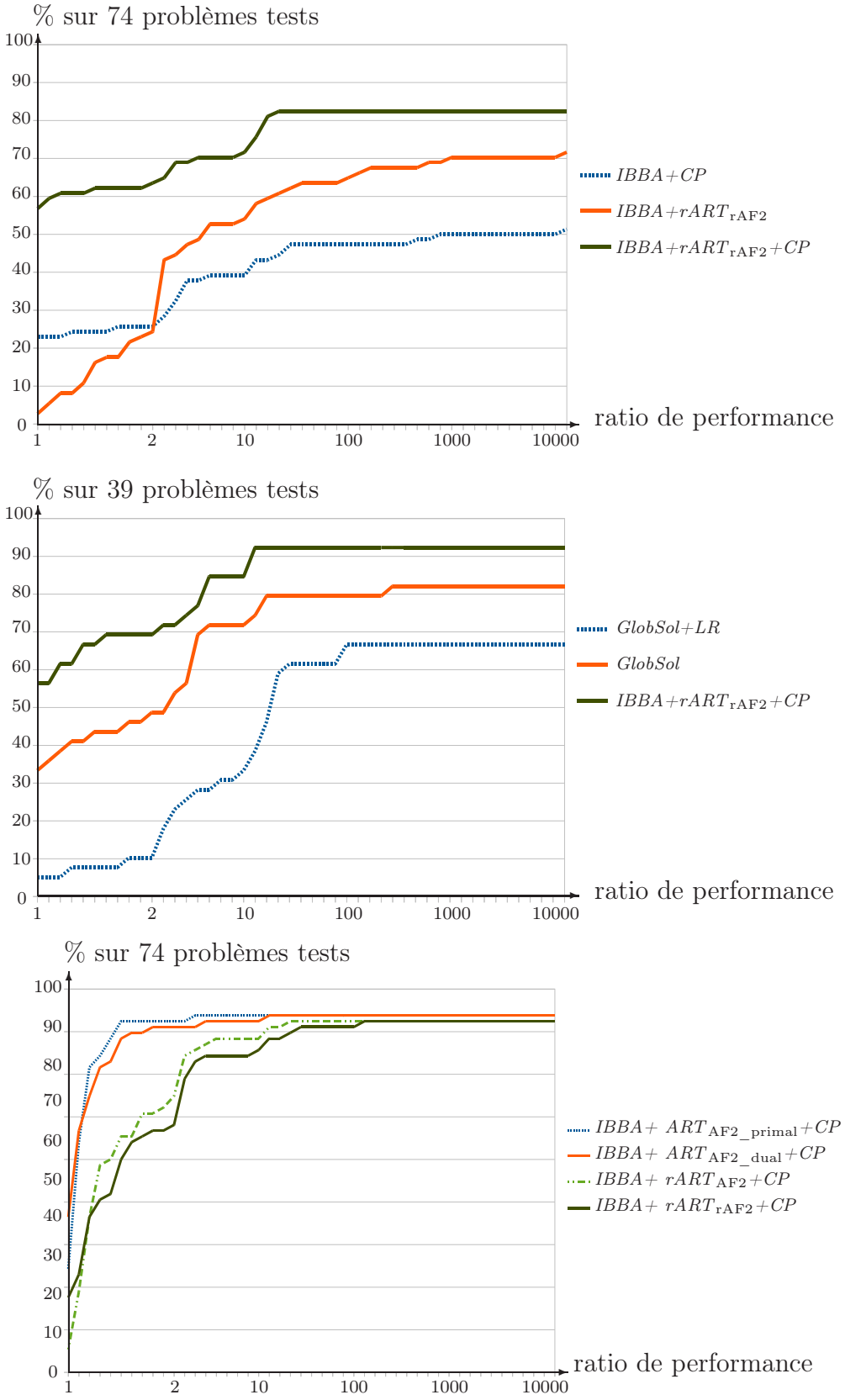


FIGURE 3.1 – Profils de performance comparant les résultats des différents algorithmes.

associé à la technique de reformulation affine robuste basée sur les formes affines rAF2 (algorithme 9) et l'arithmétique correspondante (voir section 2.5.2). Puis, $IBBA{+}rART_{\text{rAF2}}{+}CP$ désigne les résultats obtenus avec *IBBA* associé aux deux techniques d'accélération précédentes. *GlobSol+LR* et *GlobSol* représente les résultats extraits de [53] et obtenus en utilisant (ou non) la technique de relaxation linéaire basée sur la méthode RLT, voir section 1.3.3 et [54].

Les profils de performance, présentés en figure 3.1, regroupent les résultats des tableaux 3.1, 3.3 et 3.4. Il s'agit d'un outil visuel permettant de comparer les résultats de l'exécution de différents algorithmes sur un même ensemble de tests. Le pourcentage des problèmes résolus est représenté en fonction du ratio de performance, qui dépend du temps de calcul [31]. Plus précisément, pour chaque problème test, on compare le ratio entre le temps de calcul de chaque algorithme et le minimum des temps de résolution. Puis, on calcule le profil de performance, c'est-à-dire la fonction de distribution cumulative par rapport au ratio.

Remarque 3.5.1 IBBA *a aussi été testé sans technique d'accélération. Les résultats n'ont pas été ajoutés dans le tableau 3.1, car ils montraient qu'*IBBA *n'était pas très efficace sans au moins l'une des deux techniques d'accélération. Dans ce cas, seuls 24 des 74 problèmes tests ont été résolus.*

3.5.1 Validation de l'approche robuste

Dans le tableau 3.1, nous avons réalisé sur l'algorithme *IBBA* une comparaison entre la technique de propagation des contraintes, notre nouvelle technique de relaxation linéaire robuste rART et les deux combinées. Nous avons observé que :

- $IBBA{+}CP$ résout 37 problèmes tests, $IBBA{+}rART_{\text{rAF2}}$ 52 problèmes et $IBBA{+}rART_{\text{rAF2}}{+}CP$ 61 problèmes.
- Les cas résolus ne sont pas les mêmes en utilisant les deux techniques distinctes. Généralement, pour les cas non résolus, $IBBA{+}CP$ termine lorsque la limite sur le nombre d'éléments stockés dans $\mathcal{L}$ est atteinte. A l'opposé, $IBBA{+}rART_{\text{rAF2}}$ s'arrête lorsque la limite sur le temps est atteinte.
- Tous les problèmes résolus avec l'une des deux techniques d'accélération sont résolus également par l'algorithme combinant les deux techniques. De plus, celui-ci l'effectue en un temps raisonnable d'environ 1min 9s en moyenne.
- En considérant uniquement les 33 cas résolus par les 3 algorithmes, à la ligne 'Moyenne lorsque 'T' pour tous' du tableau 3.1, on remarque que le temps moyen de calcul pour $IBBA{+}CP$ est **3** fois supérieur à celui de $IBBA{+}rART_{\text{rAF2}}$, mais lorsque les deux méthodes sont combinées, il est divisé par un facteur d'environ **10** par rapport à celui de $IBBA{+}CP$ et par un facteur d'environ **3.5** par rapport à celui de $IBBA{+}rART_{\text{rAF2}}$. En considérant le nombre d'itérations, le gain de $IBBA{+}rART_{\text{rAF2}}{+}CP$ est d'environ un facteur **200** par rapport à $IBBA{+}CP$ et **3.5** par rapport à $IBBA{+}rART_{\text{rAF2}}$.

Les profils de performance de la figure 3.1 confirment que $IBBA{+}rART_{\text{rAF2}}{+}CP$ est l'algorithme le plus efficace des trois. Si l'on considère la courbe de $IBBA{+}CP$ et

IBBA+rART$_{\text{rAF2}}$, on observe que *IBBA+CP* est plus rapide que l'autre méthode, mais que *IBBA+rART*$_{\text{rAF2}}$ résout plus de problèmes, c'est pourquoi les courbes se croisent.

En observant la façon dont les deux techniques travaillent sur un sous-domaine, on remarque que *rART*$_{\text{rAF2}}$ est très efficace et la reformulation est plus précise lorsque le sous-domaine considéré est petit. Cette technique reste lente au début de l'algorithme et devient très efficace au bout d'un certain temps. A l'opposé, *CP* a un impact plus important lorsque les sous-domaines sont gros, mais vu qu'elle considère les contraintes une par une, la technique n'améliore pas trop la convergence à la fin de l'exécution de l'algorithme. C'est pourquoi la combinaison des deux techniques *CP* et *rART*$_{\text{rAF2}}$ est si efficace : *CP* réduit rapidement la taille des sous-domaines et ensuite *rART*$_{\text{rAF2}}$ améliore considérablement le minorant de chaque sous-domaine et élimine ceux qui ne contiennent pas le minimum global.

3.5.2 Comparaisons entre les différentes arithmétiques robustes de la section 2.5

Dans le tableau 3.2 et le profil de performance de la figure 3.2, nous avons utilisé l'algorithme *IBBA* (alogithme 2) combiné avec la technique de propagation des contraintes *CP* (algorithme 3) et de la technique de reformulation affine robuste *rART* (algorithme 9). Dans chacune des colonnes, nous avons utilisé l'une des trois arithmétiques affines robustes de la section 2.5 pour générer les relaxations linéaires. Le but est de montrer l'impact de l'implémentation de la robustesse en optimisation globale, car entre les trois instances, seule la façon de coder l'arithmétique affine a été modifiée.

D'après le tableau 3.2 et le profil de performance de la figure 3.2, on observe :

- L'algorithme le plus rapide est celui utilisant fAF2, les temps sont divisés par **2** par rapport à celui utilisant rAF2 et par **4.5** par rapport à celui utilisant sAF2.
- Le nombre d'itérations pour résoudre la majorité des problèmes est équivalent pour les trois algorithmes. Seuls 10 problèmes ont des nombres d'itérations très différents, cela révèle certainement une instabilité numérique durant la résolution.
- Les problèmes ex6_1_1 et ex7_2_3 ne sont résolus que par l'algorithme utilisant fAF2. Cela est dû à la vitesse de résolution ; étant donné que cet algorithme est plus rapide que les autres, il peut effectuer plus d'itérations et donc converger. Le problème ex6_2_6 n'est pas résolu par l'algorithme utilisant sAF2 pour cette même raison.
- Le profil de performance de la figure 3.2 illustre également le gain en vitesse de l'algorithme utilisant fAF2, car on peut remarquer que sa courbe domine celle des autres arithmétiques affines robustes.
- En comparant les gains grâce à fAF2 observés dans cette section et ceux observés dans le tableau 2.2 de la section 2.6, on remarque tout de même que ces gains ne sont pas aussi importants. Dans le tableau 2.2, les temps de calcul

Nom	N	M	IBBA+CP			IBBA+$rART_{\text{rAF2}}$			IBBA+$rART_{\text{rAF2}}$+CP		
			ok ?	iter	t (s)	ok ?	iter	t (s)	ok ?	iter	t (s)
hs071	4	2	**T**	9558537	722.34	**T**	1580	2.44	**T**	804	1.04
ex2_1_1	5	1	**T**	26208	1.17	**T**	151	0.32	**T**	151	0.23
ex2_1_2	6	2	**T**	105	0.00	**T**	289	0.46	**T**	105	0.18
ex2_1_3	13	9	F	2004691	106.02	**T**	352	0.74	**T**	266	0.52
ex2_1_4	6	5	**T**	5123	0.25	**T**	641	0.74	**T**	250	0.27
ex2_1_5	10	11	**T**	172195	32.70	**T**	844	1.98	**T**	263	0.66
ex2_1_6	10	5	**T**	5109625	565.22	**T**	286	0.77	**T**	285	0.69
ex2_1_7	20	10	F	7075425	1735.15	**T**	1569	16.26	**T**	1574	16.75
ex2_1_8	24	10	F	2005897	280.95	**T**	3908	53.38	**T**	1916	26.78
ex2_1_9	10	1	F	1999999	93.57	**T**	66180	160.10	**T**	60007	154.02
ex2_1_10	20	10	F	1999999	635.95	**T**	938	8.81	**T**	636	5.91
ex3_1_1	8	6	F	38000000	3604.46	**T**	81818	137.02	**T**	131195	115.92
ex3_1_2	5	6	**T**	6571	0.44	**T**	144	0.36	**T**	111	0.19
ex3_1_3	6	6	**T**	4321	0.21	**T**	243	0.55	**T**	182	0.24
ex3_1_4	3	3	**T**	21096	1.06	**T**	171	0.37	**T**	187	0.25
ex4_1_8	2	1	**T**	78417	2.31	**T**	137	0.32	**T**	128	0.11
ex4_1_9	2	2	**T**	49678	6.38	**T**	171	0.19	**T**	157	0.17
ex5_2_2_case1	9	6	F	4266494	308.90	F	2300000	3699.67	**T**	5233	8.05
ex5_2_2_case2	9	6	F	7027892	529.41	F	2200000	3646.14	**T**	9180	14.73
ex5_2_2_case3	9	6	F	3671986	257.71	F	2300000	3682.61	**T**	2255	3.44
ex5_2_4	7	6	F	3338206	510.99	**T**	128303	142.42	**T**	9848	11.30
ex5_4_2	8	6	F	43800000	3606.12	**T**	8714	12.72	**T**	201630	121.45
ex6_1_1	8	6	F	5270186	2805.03	F	1600000	3756.88	F	1500000	3775.80
ex6_1_2	4	3	**T**	15429	0.83	**T**	1813	2.39	**T**	108	0.26
ex6_1_3	12	9	F	4534626	3233.97	F	900000	3704.39	F	1000000	3913.87
ex6_1_4	6	4	F	2444266	204.92	**T**	148480	262.65	**T**	1622	2.70
ex6_2_5	9	3	F	1999999	192.80	F	800000	3934.43	F	800000	4055.02
ex6_2_6	3	1	F	2097277	124.56	F	2100000	3719.93	**T**	922664	1575.43
ex6_2_7	9	3	F	1999999	229.94	F	500000	3973.21	F	500000	4036.90
ex6_2_8	3	1	F	2003020	118.81	**T**	634377	1122.06	**T**	265276	457.87
ex6_2_9	4	2	F	3724203	369.78	F	1500000	3700.92	**T**	203775	522.57
ex6_2_10	6	3	F	1999999	241.17	F	1300000	3872.20	F	1200000	3775.14
ex6_2_11	3	1	F	2729823	149.66	**T**	214420	346.71	**T**	83487	140.51
ex6_2_12	4	2	F	2975037	202.77	**T**	1096081	2136.20	**T**	58231	112.58
ex6_2_13	6	3	F	2007671	332.47	F	1600000	3605.98	F	1500000	3650.76
ex6_2_14	4	2	**T**	8446077	988.14	**T**	450059	956.88	**T**	95170	207.78
ex7_2_1	7	14	F	9324644	2512.97	**T**	18037	50.64	**T**	8419	24.72
ex7_2_2	6	5	F	4990110	1031.30	**T**	4312	5.64	**T**	531	0.87
ex7_2_3	8	6	F	41000000	3607.35	F	2300000	3684.73	F	2200000	3716.02
ex7_2_5	5	6	**T**	6000	0.67	**T**	249	0.52	**T**	186	0.40
ex7_2_6	3	1	F	7022520	326.38	**T**	2100	1.89	**T**	1319	1.23
ex7_2_10	11	9	**T**	1417	0.09	**T**	2605	3.96	**T**	1417	2.19
ex7_3_1	4	7	**T**	33347	4.23	**T**	2713	6.21	**T**	1536	3.50
ex7_3_2	4	7	**T**	141	0.08	**T**	2831	3.05	**T**	141	0.28
ex7_3_3	5	8	**T**	18603	2.14	**T**	1104	1.74	**T**	373	0.66
ex7_3_4	12	17	F	3194446	467.81	F	800000	3756.89	F	1000000	3971.41
ex7_3_5	13	15	F	3017872	513.88	F	500000	4291.36	F	500000	4259.44
ex7_3_6	17	17	**T**	1	0.00	**T**	84	5.55	**T**	1	0.14
ex8_1_7	5	5	F	3807889	395.30	**T**	6183	12.25	**T**	1432	2.64
ex8_1_8	6	5	F	4990110	1029.01	**T**	4312	5.65	**T**	531	0.87
ex9_2_1	10	9	**T**	161	0.02	F	1800000	3637.95	**T**	64	0.26
ex9_2_2	10	11	F	5902793	314.66	F	2000000	3653.05	F	4700000	3602.48
ex9_2_3	16	15	**T**	884	0.15	F	1500000	3740.15	**T**	156	0.50
ex9_2_4	8	7	**T**	77	0.00	**T**	4682	7.96	**T**	49	0.25
ex9_2_5	8	7	**T**	51303	7.59	**T**	6331	12.69	**T**	136	0.44
ex9_2_6	16	12	F	2895007	233.85	F	1000000	3611.39	F	1200000	3756.22
ex9_2_7	10	9	**T**	161	0.02	F	1700000	3643.63	**T**	64	0.35
ex14_1_1	3	4	**T**	367	0.13	**T**	1728	2.75	**T**	301	0.54
ex14_1_2	6	9	**T**	619905	145.68	**T**	59677	206.88	**T**	24166	54.58
ex14_1_3	3	4	**T**	94	0.00	F	8000000	3629.02	**T**	91	0.26
ex14_1_5	6	6	**T**	165381	18.06	**T**	3961	6.38	**T**	1752	2.99
ex14_1_6	9	15	**T**	42139	8.88	**T**	6326	26.61	**T**	2531	12.45
ex14_1_7	10	17	F	9600000	3635.90	F	600000	4155.17	F	1100000	3703.00
ex14_1_8	3	4	**T**	98	0.01	**T**	2011	2.30	**T**	77	0.25
ex14_1_9	2	2	**T**	1300	0.05	**T**	23465	17.84	**T**	223	0.35
ex14_2_1	5	7	**T**	12017408	1683.62	**T**	30436	64.13	**T**	16786	36.73
ex14_2_2	4	5	**T**	8853	0.67	**T**	2671	3.64	**T**	1009	1.39
ex14_2_3	6	9	F	13800000	3622.13	**T**	70967	252.31	**T**	47673	173.28
ex14_2_4	5	7	**T**	1975320	455.49	**T**	62245	274.42	**T**	30002	127.56
ex14_2_5	4	5	**T**	18821	1.92	**T**	5821	11.75	**T**	2041	3.70
ex14_2_6	5	7	F	9543033	2124.91	**T**	138654	407.27	**T**	74630	237.56
ex14_2_7	6	9	F	7678896	2844.60	F	800000	4021.63	F	700000	3841.44
ex14_2_8	4	5	**T**	2085323	279.49	**T**	31840	57.17	**T**	10044	19.13
ex14_2_9	4	5	**T**	463414	70.83	**T**	19474	40.44	**T**	6582	14.59
Moyenne lorsque '**T**'			37	1 108 213.51	135.16	52	64 547.85	131.89	61	37 556.7	69.3
Moyenne lorsque '**T**' pour tous			33	1 242 503.03	151.54	33	22 023.73	52.23	33	**5 977.39**	**14.98**

TABLE 3.1 – Résultats numériques des méthodes robustes basées sur *IBBA*.

étaient divisés par **18** par rapport à ceux de rAF2, alors qu'ils ne sont divisés que par 2 une fois intégré dans rART. Néanmoins, ces gains restent tout de même conséquents et la *floating-point affine arithmetic* fAF2 est nettement la plus performante implémentation.

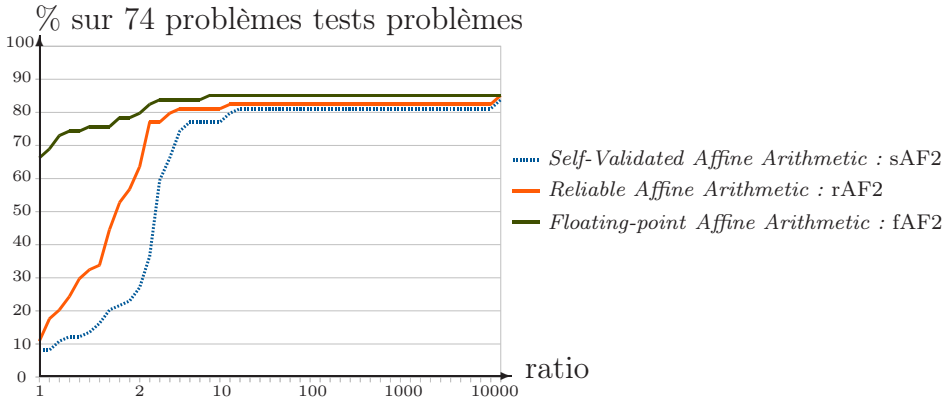


FIGURE 3.2 – Profil de performance de l'algorithme *IBBA+rART+CP* en utilisant les différentes arithmétiques affines robustes.

3.5.3 Comparaisons avec *GlobSol*

Kearfott et Hongthong dans [54] ont développé une autre technique que nous appellerons *LR* basée sur le même principe que la technique RLT, en remplaçant chaque terme non linéaire par un sous-estimateur et un sur-estimateur linéaires. Cette technique est bien connue et est déjà insérée dans le logiciel BARON [102], mais dans une version non robuste et sans arithmétique d'intervalles. Un autre article de Kearfott [53] étudie son impact lorsqu'il est intégré dans un algorithme de Branch and Bound par intervalles nommé *GlobSol* et développé par Kearfott [52]. Les résultats numériques issus de [53] ont été reportés dans le tableau 3.3 afin de les comparer avec notre algorithme *IBBA* muni des deux précédentes techniques d'accélération. Notons que le temps de calcul reporté dans le tableau dépend aussi des performances de deux ordinateurs différents, mais ceux-ci restent comparables et ne génèrent pas d'impact sur les conclusions suivantes. On observe que :

- *GlobSol+LR* résout 26 des 39 problèmes tests alors que $IBBA{+}rART_{\text{rAF2}}{+}CP$ en résout 36.
- *GlobSol* sans *LR* en résout 32.
- Dans [53], Kearfott s'est limité aux problèmes ayant au plus 10 variables. En effet, les problèmes résolus par *GlobSol* sans *LR* ont au plus 8 variables et 9 contraintes. Les problèmes résolus par $IBBA{+}rART_{\text{rAF2}}{+}CP$ ont au plus 24 variables et 17 contraintes.
- *GlobSol* sans *LR* résout 1 problème en 53 minutes alors que l'algorithme $IBBA{+}rART_{\text{rAF2}}{+}CP$ ne le résout pas en 62 minutes (ex6_1_1). En revanche, $IBBA{+}rART_{\text{rAF2}}{+}CP$ résout 5 problèmes que *GlobSol* sans *LR* ne résout pas et 10 de plus comparé à *GlobSol+LR*.

Nom	N	M	IBBA+$rART_{\text{sAF2}}$+CP			IBBA+$rART_{\text{rAF2}}$+CP			IBBA+$rART_{\text{fAF2}}$+CP		
			ok ?	iter	t (s)	ok ?	iter	t (s)	ok ?	iter	t (s)
hs071	4	2	**T**	1103	2.06	**T**	804	1.04	**T**	1103	1.12
ex2_1_1	5	1	**T**	151	0.44	**T**	151	0.23	**T**	151	0.12
ex2_1_2	6	2	**T**	105	0.39	**T**	105	0.18	**T**	105	0.08
ex2_1_3	13	9	**T**	266	1.28	**T**	266	0.52	**T**	266	0.23
ex2_1_4	6	5	**T**	231	0.49	**T**	250	0.27	**T**	224	0.18
ex2_1_5	10	11	**T**	263	1.76	**T**	263	0.66	**T**	263	0.33
ex2_1_6	10	5	**T**	285	1.72	**T**	285	0.69	**T**	285	0.26
ex2_1_7	20	10	**T**	1530	50.13	**T**	1574	16.75	**T**	1488	3.90
ex2_1_8	24	10	**T**	3507	147.02	**T**	1916	26.78	**T**	2546	5.81
ex2_1_9	10	1	**T**	83673	423.15	**T**	60007	154.02	**T**	89257	86.43
ex2_1_10	20	10	**T**	636	16.74	**T**	636	5.91	**T**	636	1.09
ex3_1_1	8	6	**T**	47180	99.52	**T**	131195	115.92	**T**	45337	50.30
ex3_1_2	5	6	**T**	111	0.40	**T**	111	0.19	**T**	111	0.10
ex3_1_3	6	6	**T**	182	0.58	**T**	182	0.24	**T**	182	0.14
ex3_1_4	3	3	**T**	182	0.42	**T**	187	0.25	**T**	179	0.15
ex4_1_8	2	1	**T**	127	0.09	**T**	128	0.11	**T**	127	0.07
ex4_1_9	2	2	**T**	157	0.41	**T**	157	0.17	**T**	157	0.13
ex5_2_2_case1	9	6	**T**	5254	15.26	**T**	5233	8.05	**T**	5187	6.17
ex5_2_2_case2	9	6	**T**	9195	26.55	**T**	9180	14.73	**T**	9120	11.54
ex5_2_2_case3	9	6	**T**	2115	6.29	**T**	2255	3.44	**T**	2165	2.52
ex5_2_4	7	6	**T**	9842	18.38	**T**	9848	11.30	**T**	9787	8.46
ex5_4_2	8	6	**T**	4241	8.73	**T**	201630	121.45	**T**	27599	15.15
ex6_1_1	8	6	F	1000000	3791.47	F	1500000	3775.80	**T**	1725033	3161.74
ex6_1_2	4	3	**T**	108	0.36	**T**	108	0.26	**T**	108	0.11
ex6_1_3	12	9	F	500000	3940.63	F	1000000	3913.87	F	1500000	3740.83
ex6_1_4	6	4	**T**	1622	4.57	**T**	1622	2.70	**T**	1622	2.87
ex6_2_5	9	3	F	200000	3657.01	F	800000	4055.02	F	1999999	1911.03
ex6_2_6	3	1	F	900000	3622.02	**T**	922664	1575.43	**T**	921910	927.98
ex6_2_7	9	3	F	200000	5759.67	F	500000	4036.90	F	1600000	3663.47
ex6_2_8	3	1	**T**	265316	1071.23	**T**	265276	457.87	**T**	265318	269.43
ex6_2_9	4	2	**T**	204272	1291.05	**T**	203775	522.57	**T**	204240	250.83
ex6_2_10	6	3	F	400000	4014.02	F	1200000	3775.14	F	4200000	3657.38
ex6_2_11	3	1	**T**	83488	285.68	**T**	83487	140.51	**T**	83481	81.33
ex6_2_12	4	2	**T**	58231	259.33	**T**	58231	112.58	**T**	58231	58.10
ex6_2_13	6	3	F	500000	3634.90	F	1500000	3650.76	F	4287127	3027.83
ex6_2_14	4	2	**T**	95166	456.06	**T**	95170	207.78	**T**	95945	116.37
ex7_2_1	7	14	**T**	8416	50.63	**T**	8419	24.72	**T**	8415	15.06
ex7_2_2	6	5	**T**	531	1.19	**T**	531	0.87	**T**	531	0.79
ex7_2_3	8	6	F	1300000	3665.27	F	2200000	3716.02	**T**	2486037	2797.47
ex7_2_5	5	6	**T**	186	0.67	**T**	186	0.40	**T**	186	3.51
ex7_2_6	3	1	**T**	1319	1.54	**T**	1319	1.23	**T**	1319	1.20
ex7_2_10	11	9	**T**	1417	4.02	**T**	1417	2.19	**T**	1417	1.19
ex7_3_1	4	7	**T**	1536	6.83	**T**	1536	3.50	**T**	1536	2.02
ex7_3_2	4	7	**T**	141	0.48	**T**	141	0.28	**T**	141	0.32
ex7_3_3	5	8	**T**	496	1.19	**T**	373	0.66	**T**	496	0.77
ex7_3_4	12	17	F	400000	4008.88	F	1000000	3971.41	F	2700000	3622.91
ex7_3_5	13	15	F	200000	4495.04	F	500000	4259.44	F	2200000	3717.38
ex7_3_6	17	17	**T**	1	0.21	**T**	1	0.14	**T**	1	0.28
ex8_1_7	5	5	**T**	1434	4.69	**T**	1432	2.64	**T**	1437	1.63
ex8_1_8	6	5	**T**	531	1.23	**T**	531	0.87	**T**	531	0.81
ex9_2_1	10	9	**T**	64	0.21	**T**	64	0.26	**T**	64	0.28
ex9_2_2	10	11	F	2900000	3701.73	F	4700000	3602.48	F	5902793	3500.72
ex9_2_3	16	15	**T**	161	1.53	**T**	156	0.50	**T**	161	0.43
ex9_2_4	8	7	**T**	49	0.11	**T**	49	0.25	**T**	49	0.28
ex9_2_5	8	7	**T**	159	0.37	**T**	136	0.44	**T**	159	0.41
ex9_2_6	16	12	F	600000	3778.58	F	1200000	3756.22	F	2430385	3443.14
ex9_2_7	10	9	**T**	64	1.18	**T**	64	0.35	**T**	64	1.33
ex14_1_1	3	4	**T**	300	0.60	**T**	301	0.54	**T**	300	0.52
ex14_1_2	6	9	**T**	23976	134.43	**T**	24166	54.58	**T**	23976	22.44
ex14_1_3	3	4	**T**	91	0.13	**T**	91	0.26	**T**	91	0.31
ex14_1_5	6	6	**T**	2973	5.70	**T**	1752	2.99	**T**	2218	2.94
ex14_1_6	9	15	**T**	2338	25.42	**T**	2531	12.45	**T**	2577	6.57
ex14_1_7	10	17	F	500000	4017.37	F	1100000	3703.00	F	3500000	3695.28
ex14_1_8	3	4	**T**	77	1.17	**T**	77	0.25	**T**	77	0.35
ex14_1_9	2	2	**T**	223	0.24	**T**	223	0.35	**T**	242	0.41
ex14_2_1	5	7	**T**	16804	78.12	**T**	16786	36.73	**T**	16829	22.60
ex14_2_2	4	5	**T**	1018	1.97	**T**	1009	1.39	**T**	1018	1.15
ex14_2_3	6	9	**T**	47896	440.32	**T**	47673	173.28	**T**	47842	87.84
ex14_2_4	5	7	**T**	29946	287.35	**T**	30002	127.56	**T**	29936	47.16
ex14_2_5	4	5	**T**	2041	6.56	**T**	2041	3.70	**T**	2041	2.15
ex14_2_6	5	7	**T**	74672	643.02	**T**	74630	237.56	**T**	74672	121.59
ex14_2_7	6	9	F	300000	5257.87	F	700000	3841.44	F	1500000	3623.26
ex14_2_8	4	5	**T**	10106	45.02	**T**	10044	19.13	**T**	10107	11.91
ex14_2_9	4	5	**T**	6581	34.81	**T**	6582	14.59	**T**	6582	8.53
Moyenne lorsque '**T**'			60	18568.1	99.52	61	37556.7	69.30	63	99574.84	130.59
Moyenne lorsque '**T**' pour tous			60	18568.1	99.52	60	22804.92	44.2	60	19003.92	22.33

TABLE 3.2 – Résultats numériques de l'algorithme IBBA+$rART$+CP en utilisant les trois arithmétiques affines robustes de la section 2.5.

Nom	N	M	Globsol + LR		GlobSol		IBBA+ CP + rART$_{rAF2}$	
			ok ?	t (min)	ok ?	t (min)	ok ?	t (min)
hs071	4	2					**T**	0.0174
ex2_1_1	5	1	**T**	0.0930	**T**	0.0180	**T**	0.0039
ex2_1_2	6	2	**T**	0.0768	**T**	0.0633	**T**	0.0031
ex2_1_4	6	5	**T**	0.1640	**T**	0.0855	**T**	0.0046
ex3_1_1	8	6	F	60.1775	F	60.1745	**T**	1.9320
ex3_1_4	3	3	**T**	0.0038	**T**	0.0030	**T**	0.0042
ex4_1_8	2	1	**T**	0.0118	**T**	0.0003	**T**	0.0019
ex4_1_9	2	2	**T**	0.0067	**T**	0.0097	**T**	0.0029
ex5_2_4	7	6	F	60.2352	F	43.9670	**T**	0.1884
ex5_4_2	8	6	**T**	1.3218	**T**	4.9393	**T**	2.0242
ex6_1_1	8	6	F	100.9517	**T**	53.3900	F	62.9300
ex6_1_2	4	3	**T**	0.4143	**T**	0.0242	**T**	0.0043
ex6_1_4	6	4	**T**	4.4903	**T**	0.2373	**T**	0.0450
ex6_2_6	3	1	F	60.2438	**T**	5.0968	**T**	26.2572
ex6_2_8	3	1	F	60.0058	**T**	3.4003	**T**	7.6311
ex6_2_9	4	2	F	60.0280	**T**	7.7175	**T**	8.7095
ex6_2_10	6	3	F	60.0987	F	60.0880	F	62.9190
ex6_2_11	3	1	F	60.0127	**T**	4.5502	**T**	2.3419
ex6_2_12	4	2	F	60.0252	**T**	3.2658	**T**	1.8764
ex6_2_13	6	3	F	60.0785	F	60.0702	F	60.8459
ex6_2_14	4	2	**T**	10.4143	**T**	0.5315	**T**	3.4630
ex7_2_2	6	5	**T**	0.4575	**T**	0.0867	**T**	0.0144
ex7_2_5	5	6	**T**	0.2740	**T**	0.0383	**T**	0.0067
ex7_2_6	3	1	**T**	0.0103	**T**	0.0033	**T**	0.0205
ex7_3_1	4	7	**T**	0.1962	**T**	0.0443	**T**	0.0584
ex7_3_2	4	7	**T**	0.0043	**T**	0.0062	**T**	0.0047
ex7_3_3	5	8	**T**	0.1855	**T**	0.0557	**T**	0.0109
ex8_1_7	5	5	F	60.0090	F	60.0070	**T**	0.0440
ex8_1_8	6	5	**T**	0.4620	**T**	0.0870	**T**	0.0146
ex9_2_4	8	7	F	62.8002	F	64.0362	**T**	0.0042
ex9_2_5	8	7	F	61.1955	F	39.0192	**T**	0.0073
ex14_1_1	3	4	**T**	0.1883	**T**	0.2037	**T**	0.0090
ex14_1_2	6	9	**T**	0.2137	**T**	0.0803	**T**	0.9097
ex14_1_3	3	4	**T**	0.1293	**T**	1.7860	**T**	0.0043
ex14_1_5	6	6	**T**	0.1933	**T**	0.0548	**T**	0.0499
ex14_1_9	2	2	**T**	0.0132	**T**	0.0075	**T**	0.0059
ex14_2_1	5	7	**T**	0.5452	**T**	0.0550	**T**	0.6122
ex14_2_2	4	5	**T**	0.0100	**T**	0.0057	**T**	0.0232
ex14_2_3	6	9	**T**	1.3413	**T**	0.1728	**T**	2.8881
ex14_2_5	4	5	**T**	0.3235	**T**	0.0715	**T**	0.0617
Moyenne lorsque '**T**'			26	0.83	32	2.69	36	1.65

TABLE 3.3 – Comparaison entre l'approche de *GlobSol* [53] et notre approche *rART*.

3.5.4 Comparaisons avec les méthodes non fiables

Dans le tableau 3.4, les résultats des algorithmes d'optimisation globale non fiables et non robustes sont présentés pour évaluer le coût de la robustesse dans notre algorithme combinant les techniques *CP* et ART_{AF2}. Ainsi, nous avons testé dans trois cas *IBBA* (algorithme 2) en intégrant ou non la technique *CP* et la technique ART_{AF2} (algorithme 8) lorsque l'arithmétique affine AF2 n'est pas strictement fiable, voir section 2.5.

Dans la première et deuxième colonne, *ART* (algorithme 8) est utilisé et le programme linéaire associé pour le calcul du minorant est résolu en utilisant la formulation primale du programme (P2) pour la colonne $ART_{\text{AF2\_primal}}$ et la formulation duale pour la colonne $ART_{\text{AF2\_dual}}$. Dans la troisième colonne $rART_{\text{AF2}}$, nous utilisons *rART* (algorithme 9) mais le programme linéaire est généré directement avec la forme affine AF2 à la place de rAF2, ainsi le programme linéaire n'est pas complètement fiable. On observe que :

– 1 problème test ex9_2_2 qui n'a pas été résolu par *IBBA*+$rART_{\text{AF2}}$ +*CP* a pu être résolu par la version non robuste de notre code. Ceci est dû au fait que pour cet exemple la recherche d'une solution réalisable n'est pas facile. Or, la version non robuste de notre méthode permet de tester aussi le point

solution de la relaxation $\mathcal{T}^{-1}(z_{sol})$, et non seulement le point milieu comme dans la version robuste. C'est grâce à ce test supplémentaire qu'une solution réalisable est trouvée. A la fin de l'exécution des versions robustes du code, le minorant certifié est bon, mais puisqu'aucune solution réalisable n'est trouvée, l'algorithme ne termine pas.

- Pour les autres problèmes ayant environ le même nombre de variables que de contraintes, des résultats similaires sont obtenus en utilisant les formulations primales et duales.
- En comparant les temps de calcul sur les 61 problèmes résolus par les trois méthodes, on remarque que l'introduction du calcul fiable et robuste des minorants et certificats d'infaisabilité proposée par la méthode de Neumaier et Shcherbina [82] nécessite **14**% de temps supplémentaire.
- En utilisant les versions non fiables de l'algorithme, **1** problème supplémentaire (ex9_2_2) est résolu, mais il faut tout de même 5 minutes pour le résoudre. En moyenne, la version entièrement robuste $IBBA{+}rART_{\text{rAF2}}\ {+}CP$ nécessite **1.3** fois plus de temps de calcul que les trois versions non robustes. En effet, le temps moyen de la méthode robuste est de 69.3 secondes sur les 61 problèmes résolus par les quatre algorithmes, comparé à environ 55 secondes pour les trois versions non robustes. En revanche, la version entièrement robuste résout les problèmes en **1.4** fois moins d'itérations que les versions non robustes : 37556.7 itérations en moyenne sur les 61 problèmes pour $IBBA{+}rART_{\text{rAF2}}\ {+}CP$ et environ 55000 itérations pour les trois versions du tableau 3.4.
- Toutes les méthodes présentées dans le tableau 3.4 restent très efficaces : aucune erreur numérique n'aboutit à une fausse solution optimale (toutes les solutions sont certifiées).

En observant encore les profils de performance de la figure 3.1, les comportements des différentes versions des algorithmes sont pratiquement tous équivalents à ceux de $IBBA{+}rART_{\text{rAF2}}{+}CP$.

3.5.5 Test de la méthode mixte

Afin de comparer les techniques ART et MART, nous les avons incluses dans notre algorithme *IBBA*. Nécessairement, l'algorithme utilisant MART doit visiter moins d'éléments que la méthode basée sur ART. Ceci est simplement dû au fait que ART et MART génèrent le même programme linéaire. La seule différence est que MART recentre les variables entières dans leur espace d'origine pour produire un programme linéaire mixte. Mais la résolution de ce programme linéaire mixte peut nécessiter beaucoup plus de temps que celle du programme linéaire. En effet, la plupart des algorithmes de résolution des programmes linéaires mixtes incluent un autre algorithme de Branch and Bound. Ainsi, nous obtenons un algorithme de Branch and Bound à chaque itération de notre propre algorithme de Branch and Bound par intervalles, et non plus simplement une résolution par un algorithme du simplexe ou de point intérieur, en utilisant la relaxation continue de la technique ART.

Nom	N	M	IBBA+$ART_{\text{AF2\_primal}}$+CP			IBBA+$ART_{\text{AF2\_dual}}$+CP			IBBA+$rART_{\text{AF2}}$+CP		
			ok ?	iter	t (s)	ok ?	iter	t (s)	ok ?	iter	t (s)
hs071	4	2	**T**	700	1.03	**T**	701	1.06	**T**	809	1.04
ex2_1_1	5	1	**T**	8	0.25	**T**	8	0.19	**T**	151	0.27
ex2_1_2	6	2	**T**	1	0.21	**T**	1	0.20	**T**	105	0.24
ex2_1_3	13	9	**T**	3	0.18	**T**	3	0.28	**T**	266	0.42
ex2_1_4	6	5	**T**	1	0.16	**T**	1	0.15	**T**	245	0.39
ex2_1_5	10	11	**T**	73	0.24	**T**	73	0.23	**T**	268	0.49
ex2_1_6	10	5	**T**	9	0.19	**T**	9	0.19	**T**	285	0.50
ex2_1_7	20	10	**T**	792	1.45	**T**	792	1.45	**T**	1574	3.54
ex2_1_8	24	10	**T**	133	0.24	**T**	133	0.22	**T**	3359	6.78
ex2_1_9	10	1	**T**	37810	39.74	**T**	37791	36.06	**T**	60007	52.97
ex2_1_10	20	10	**T**	277	0.56	**T**	277	0.57	**T**	640	1.11
ex3_1_1	8	6	**T**	35533	41.32	**T**	35470	36.63	**T**	55492	53.73
ex3_1_2	5	6	**T**	59	0.21	**T**	59	0.25	**T**	111	0.24
ex3_1_3	6	6	**T**	106	0.27	**T**	106	0.28	**T**	182	0.28
ex3_1_4	3	3	**T**	92	0.27	**T**	92	0.28	**T**	183	0.28
ex4_1_8	2	1	**T**	57	0.16	**T**	57	0.17	**T**	127	0.21
ex4_1_9	2	2	**T**	139	0.25	**T**	139	0.31	**T**	157	0.29
ex5_2_2_case1	9	6	**T**	4941	6.70	**T**	4941	5.73	**T**	5353	6.38
ex5_2_2_case2	9	6	**T**	8983	11.99	**T**	8983	10.51	**T**	9307	11.81
ex5_2_2_case3	9	6	**T**	1846	2.49	**T**	1846	2.25	**T**	2211	2.68
ex5_2_4	7	6	**T**	9440	8.08	**T**	9437	8.11	**T**	9914	8.47
ex5_4_2	8	6	**T**	3680	3.84	**T**	3704	3.66	**T**	137661	59.64
ex6_1_1	8	6	F	2000000	3794.47	F	2200000	3703.80	F	2000000	3622.57
ex6_1_2	4	3	**T**	138	0.30	**T**	138	0.37	**T**	138	0.32
ex6_1_3	12	9	F	1400000	3871.08	F	1600000	3628.10	F	1600000	3816.64
ex6_1_4	6	4	**T**	2726	3.43	**T**	2727	3.02	**T**	2748	3.24
ex6_2_5	9	3	F	1999999	2005.80	F	1999999	1952.08	F	1999999	1925.24
ex6_2_6	3	1	**T**	1538181	1285.85	**T**	1538312	1270.75	**T**	1543008	1394.78
ex6_2_7	9	3	F	1800000	3637.42	F	1999999	3725.52	F	1900000	3621.72
ex6_2_8	3	1	**T**	443676	420.40	**T**	442948	384.56	**T**	446971	413.38
ex6_2_9	4	2	**T**	334939	342.10	**T**	334963	355.23	**T**	337766	368.95
ex6_2_10	6	3	F	4333634	2931.14	F	4333634	2978.80	F	4333400	3024.25
ex6_2_11	3	1	**T**	135634	124.80	**T**	135635	119.00	**T**	135891	125.55
ex6_2_12	4	2	**T**	92057	76.53	**T**	92057	81.32	**T**	92837	86.22
ex6_2_13	6	3	F	4170289	2362.08	F	4170289	2415.73	F	4170149	2492.90
ex6_2_14	4	2	**T**	99275	90.03	**T**	99282	97.22	**T**	134351	145.79
ex7_2_1	7	14	**T**	10756	18.11	**T**	10756	17.43	**T**	12418	21.24
ex7_2_2	6	5	**T**	583	0.76	**T**	583	0.84	**T**	558	0.76
ex7_2_3	8	6	F	4700000	3684.41	F	4700000	3600.61	F	4500000	3624.55
ex7_2_5	5	6	**T**	322	0.47	**T**	322	0.57	**T**	361	0.50
ex7_2_6	3	1	**T**	2083	1.75	**T**	2083	1.72	**T**	2566	2.08
ex7_2_10	11	9	**T**	37	0.22	**T**	37	0.28	**T**	1417	1.32
ex7_3_1	4	7	**T**	1506	2.28	**T**	1506	1.72	**T**	1536	1.92
ex7_3_2	4	7	**T**	82	0.24	**T**	82	0.31	**T**	141	0.29
ex7_3_3	5	8	**T**	204	0.42	**T**	204	0.51	**T**	373	0.59
ex7_3_4	12	17	F	3414184	3465.06	F	3600000	3681.48	F	3275442	3673.87
ex7_3_5	13	15	F	3439246	3479.46	F	3072081	3378.28	F	2228770	3417.18
ex7_3_6	17	17	**T**	1	0.16	**T**	1	0.20	**T**	1	0.15
ex8_1_7	5	5	**T**	1115	1.46	**T**	1107	1.38	**T**	1439	1.53
ex8_1_8	6	5	**T**	583	0.82	**T**	583	0.85	**T**	558	0.78
ex9_2_1	10	9	**T**	64	0.23	**T**	64	0.24	**T**	64	0.26
ex9_2_2	10	11	**T**	589806	329.03	**T**	589806	331.08	F	5902793	3451.84
ex9_2_3	16	15	**T**	35	0.22	**T**	35	0.21	**T**	156	0.43
ex9_2_4	8	7	**T**	49	0.30	**T**	49	0.23	**T**	49	0.22
ex9_2_5	8	7	**T**	133	0.33	**T**	133	0.30	**T**	136	0.39
ex9_2_6	16	12	F	2800000	3733.08	F	3200000	3676.93	F	2476882	3446.02
ex9_2_7	10	9	**T**	64	0.24	**T**	64	0.25	**T**	64	0.26
ex14_1_1	3	4	**T**	2	0.17	**T**	185	0.78	**T**	300	0.46
ex14_1_2	6	9	**T**	23769	20.25	**T**	23777	19.45	**T**	23976	20.69
ex14_1_3	3	4	**T**	73	0.22	**T**	66	0.25	**T**	94	0.26
ex14_1_5	6	6	**T**	629	1.06	**T**	665	1.03	**T**	2996	3.61
ex14_1_6	9	15	**T**	2	0.31	**T**	1484	3.12	**T**	2266	5.38
ex14_1_7	10	17	F	3900000	3660.68	F	3900000	3705.58	F	3800000	3651.52
ex14_1_8	3	4	**T**	73	0.27	**T**	73	0.22	**T**	94	0.26
ex14_1_9	2	2	**T**	220	0.35	**T**	220	0.37	**T**	821	0.75
ex14_2_1	5	7	**T**	33377	40.10	**T**	33362	38.10	**T**	138055	147.07
ex14_2_2	4	5	**T**	1292	1.14	**T**	1437	1.26	**T**	2212	1.84
ex14_2_3	6	9	**T**	112983	176.29	**T**	112721	166.45	**T**	119849	178.00
ex14_2_4	5	7	**T**	195226	190.41	**T**	72572	81.95	**T**	51533	63.20
ex14_2_5	4	5	**T**	2563	2.31	**T**	2621	2.27	**T**	2581	2.35
ex14_2_6	5	7	**T**	171158	238.47	**T**	152460	210.87	**T**	171690	231.39
ex14_2_7	6	9	F	1900000	3792.93	F	1900000	3773.64	F	1800000	3608.86
ex14_2_8	4	5	**T**	18853	19.03	**T**	18853	18.89	**T**	23403	23.19
ex14_2_9	4	5	**T**	13194	14.22	**T**	12789	13.83	**T**	16211	17.45
Moyenne lorsque '**T**'			62	63421.71	56.85	62	61151.37	53.83	61	58361.23	57.03
Moyenne lorsque '**T**' pour tous			61	54792.46	52.39	61	52484.9	49.28	61	**58361.23**	**57.03**

TABLE 3.4 – Résultats numériques pour les méthodes d'optimisation globale non robustes, mais exactes.

Pour illustrer l'application de ces deux méthodes, nous avons testé l'algorithme sur un problème d'optimisation globale non convexe de la librairie 1 de COCONUT [81], nommé $ex7\_2\_1$. Ce problème a 7 variables et 14 contraintes. Nous avons juste changé le type des variables x_1, x_2, x_4 et x_5 afin qu'il soit considéré comme des variables entières.

L'algorithme *IBBA* incluant MART résout le problème en **9 046** itérations et **43** secondes ; l'algorithme *IBBA* incluant ART le résout en **45 598** itérations et **49** secondes. Sur ce simple petit exemple, on peut remarquer que le nombre d'itérations est divisé par **5**, mais le temps de calcul par itération est pratiquement multiplié par **5**.

Ces premiers résultats tendraient à montrer que la technique MART pourrait avoir un impact plus important lorsque les problèmes à traiter sont de taille importante et difficile à résoudre par notre algorithme de Branch and Bound par intervalles.

3.5.6 Résultats complémentaires

Dans le tableau 3.5, la colonne 'notre UB' correspond au meilleur majorant du minimum global trouvé, et la colonne 'UB de COCONUT' correspond à celui répertorié sur le site COCONUT [81] et trouvé par l'algorithme dont le nom est inscrit dans la colonne 'Algorithme' du tableau. On remarque que nos majorants sont très proches de ceux de COCONUT. Ces petites différences proviennent de la précision que nous garantissons sur les contraintes, qui est de 10^{-8}.

Notre algorithme *IBBA* muni de la technique d'accélération $rART_{\text{rAF2}}$ ou de la version non robuste ART_{AF2}, a montré son intérêt dans les sections précédentes. Néanmoins, il reste 13 problèmes tests encore non résolus par notre approche. Ceci est en partie imputable à la limite imposée sur le temps de calcul (moins d'une heure) et à la limite sur la mémoire disponible (moins de 2 millions d'éléments stockés dans $\mathcal{L}$), mais pas seulement.

Dans notre code *IBBA*, le majorant de la fonction objectif (c'est-à-dire notre minimum courant) est obtenu par simple évaluation au point milieu. Cette heuristique est simple, mais insuffisante pour certains problèmes dont le domaine des solutions réalisables est très réduit. Afin d'améliorer notre algorithme de Branch and Bound par intervalles, il serait donc judicieux d'effectuer quelque pas de descente d'un algorithme d'optimisation locale. Ceci n'étant pas l'objet de notre étude, nous n'avons pas développé cette idée pour nous concentrer sur l'impact de nos méthodes de reformulation. Néanmoins, l'idée de tester le point solution de la relaxation linéaire a été incluse dans les versions $IBBA{+}ART_{\text{AF2\_primal}}{+}CP$ et $IBBA{+}ART_{\text{AF2\_dual}}{+}CP$. Ce test a montré son efficacité, car plus de la moitié des mises à jour du minimum courant ont été obtenues grâce à ce test. Notamment, le test *ex9_2_2* est résolu avec ces deux versions grâce à la solution réalisable trouvée en testant ce point.

Pour les autres versions et pour palier ce problème de recherche de bonnes solutions réalisables, nous avons relancé l'optimisation sur les 13 problèmes tests res-

Nom	N	M	ok ?	iter	t (s)	max $\|\mathcal{L}\|$	Notre UB	UB de [81]	Algorithme
hs071	4	2	**T**	804	1.04	735	17.014017363	17.0140	DONLP2
ex2_1_1	5	1	**T**	151	0.23	20	-16.999999870	-17.0000	BARON 7.2
ex2_1_2	6	2	**T**	105	0.18	105	-212.999999704	-213.0000	MINOS
ex2_1_3	13	9	**T**	266	0.52	260	-14.999999864	-15.0000	BARON 7.2
ex2_1_4	6	5	**T**	250	0.27	223	-10.999999934	-11.0000	DONLP2
ex2_1_5	10	11	**T**	263	0.66	203	-268.014631487	-268.0146	MINOS
ex2_1_6	10	5	**T**	285	0.69	40	-38.999999657	-39.0000	BARON 7.2
ex2_1_7	20	10	**T**	1574	16.75	1142	-4150.410133579	-4150.4101	BARON 7.2
ex2_1_8	24	10	**T**	1916	26.78	1916	15639.000022211	15639.0000	BARON 7.2
ex2_1_9	10	1	**T**	60007	154.02	25762	-0.375000000	-0.3750	MINOS
ex2_1_10	20	10	**T**	636	5.91	636	49318.017963635	49318.0180	MINOS
ex3_1_1	8	6	**T**	131195	115.92	17898	7049.248020538	7049.2083	BARON 7.2
ex3_1_2	5	6	**T**	111	0.19	44	-30665.538672616	-30665.5400	LINGO8
ex3_1_3	6	6	**T**	182	0.24	133	-309.999998195	-310.0000	BARON 7.2
ex3_1_4	3	3	**T**	187	0.25	23	-3.999999985	-4.0000	DONLP2
ex4_1_8	2	1	**T**	128	0.11	61	-16.738893157	-16.7389	MINOS
ex4_1_9	2	2	**T**	157	0.17	15	-5.508013267	-5.5080	BARON 7.2
ex5_2_2_case1	9	6	**T**	5233	8.05	2634	-399.999999744	-400.0000	DONLP2
ex5_2_2_case2	9	6	**T**	9180	14.73	3681	-599.999999816	-600.0000	BARON 7.2
ex5_2_2_case3	9	6	**T**	2255	3.44	1534	-749.999999952	-750.0000	DONLP2
ex5_2_4	7	6	**T**	9848	11.30	9076	-449.999999882	-450.0000	MINOS
ex5_4_2	8	6	**T**	201630	121.45	9324	7512.230144503	7512.2259	BARON 7.2
ex6_1_1	8	6	F	1500000	3775.80	379077	$+\infty$	-0.0202	BARON 7.2
ex6_1_2	4	3	**T**	108	0.26	46	-0.032463785	-0.0325	BARON 7.2
ex6_1_3	12	9	F	1000000	3913.87	365054	$+\infty$	-0.3525	BARON 7.2
ex6_1_4	6	4	**T**	1622	2.70	692	-0.294541288	-0.2945	MINOS
ex6_2_5	9	3	F	800000	4055.02	800000	$+\infty$	-70.7521	MINOS
ex6_2_6	3	1	**T**	922664	1575.43	81602	-0.000002603	0.0000	DONLP2
ex6_2_7	9	3	F	500000	4036.90	500000	$+\infty$	-0.1608	BARON 7.2
ex6_2_8	3	1	**T**	265276	457.87	18364	-0.027006349	-0.0270	BARON 7.2
ex6_2_9	4	2	**T**	203775	522.57	15583	-0.034066184	-0.0341	MINOS
ex6_2_10	6	3	F	1200000	3775.14	697120	-3.051949753	-3.0520	BARON 7.2
ex6_2_11	3	1	**T**	83487	140.51	6996	-0.000002672	0.0000	MINOS
ex6_2_12	4	2	**T**	58231	112.58	4096	0.289194748	0.2892	BARON 7.2
ex6_2_13	6	3	F	1500000	3650.76	797908	-0.216206601	-0.2162	BARON 7.2
ex6_2_14	4	2	**T**	95170	207.78	16715	-0.695357929	-0.6954	MINOS
ex7_2_1	7	14	**T**	8419	24.72	5288	1227.226078824	1227.1896	BARON 7.2
ex7_2_2	6	5	**T**	531	0.87	469	-0.388811439	-0.3888	DONLP2
ex7_2_3	8	6	F	2200000	3716.02	107484	7049.277305603	7049.2181	MINOS
ex7_2_5	5	6	**T**	186	0.40	85	10122.493318794	10122.4828	BARON 7.2
ex7_2_6	3	1	**T**	1319	1.23	319	-83.249728842	-83.2499	BARON 7.2
ex7_2_10	11	9	**T**	1417	2.19	1262	0.100000006	0.1000	MINOS
ex7_3_1	4	7	**T**	1536	3.50	705	0.341739562	0.3417	BARON 7.2
ex7_3_2	4	7	**T**	141	0.28	55	1.089863971	1.0899	DONLP2
ex7_3_3	5	8	**T**	373	0.66	206	0.817529051	0.8175	BARON 7.2
ex7_3_4	12	17	F	1000000	3971.41	585427	$+\infty$	6.2746	BARON 7.2
ex7_3_5	13	15	F	500000	4259.44	458420	$+\infty$	1.2036	BARON 7.2
ex7_3_6	17	17	**T**	1	0.14	0	$+\infty$	0.0000	MINOS
ex8_1_7	5	5	**T**	1432	2.64	1153	0.029310832	0.0293	MINOS
ex8_1_8	6	5	**T**	531	0.87	469	-0.388811439	-0.3888	DONLP2
ex9_2_1	10	9	**T**	64	0.26	3	17.000000000	17.0000	DONLP2
ex9_2_2	10	11	F	4700000	3602.48	1568227	$+\infty$	99.9995	DONLP2
ex9_2_3	16	15	**T**	156	0.50	11	0	0.0000	MINOS
ex9_2_4	8	7	**T**	49	0.25	5	0.500000000	0.5000	DONLP2
ex9_2_5	8	7	**T**	136	0.44	56	5.000000026	5.0000	MINOS
ex9_2_6	16	12	F	1200000	3756.22	963638	56.320312500	-1.0000	DONLP2
ex9_2_7	10	9	**T**	64	0.35	3	17.000000000	17.0000	DONLP2
ex14_1_1	3	4	**T**	301	0.54	15	-9.00E-09	0.0000	MINOS
ex14_1_2	6	9	**T**	24166	54.58	7294	2.00E-09	0.0000	MINOS
ex14_1_3	3	4	**T**	91	0.26	20	8.00E-09	0.0000	BARON 7.2
ex14_1_5	6	6	**T**	1752	2.99	340	7.00E-09	0.0000	DONLP2
ex14_1_6	9	15	**T**	2531	12.45	380	8.00E-09	0.0000	DONLP2
ex14_1_7	10	17	F	1100000	3703.00	8556	3234.994301063	0.0000	BARON 7.2
ex14_1_8	3	4	**T**	77	0.25	23	9.00E-09	0.0000	BARON 7.2
ex14_1_9	2	2	**T**	223	0.35	36	-4.00E-09	0.0000	MINOS
ex14_2_1	5	7	**T**	16786	36.73	2559	9.00E-09	0.0000	MINOS
ex14_2_2	4	5	**T**	1009	1.39	160	9.00E-09	0.0000	MINOS
ex14_2_3	6	9	**T**	47673	173.28	9706	4.00E-09	0.0000	MINOS
ex14_2_4	5	7	**T**	30002	127.56	8758	9.00E-09	0.0000	MINOS
ex14_2_5	4	5	**T**	2041	3.70	188	9.00E-09	0.0000	MINOS
ex14_2_6	5	7	**T**	74630	237.56	24002	4.00E-09	0.0000	MINOS
ex14_2_7	6	9	F	700000	3841.44	256647	0.218278728	0.0000	MINOS
ex14_2_8	4	5	**T**	10044	19.13	358	9.00E-09	0.0000	MINOS
ex14_2_9	4	5	**T**	6582	14.59	603	9.00E-09	0.0000	MINOS
Moyenne lorsque '**T**'			61	37556.7	69.30	4657.21			

TABLE 3.5 – Comparaison des solutions données par [81] et celles obtenues avec $IBBA{+}rART_{\text{rAF2}}{+}CP$.

tant, et ce, en donnant dès le début de l'algorithme la meilleure solution connue, c'est-à-dire qu'on initialise $\tilde{f}$ avec cette solution. Celles-ci sont disponibles sur le site COCONUT [81] et sont répertoriées dans la colonne 'UB de [81]' du tableau 3.3, la colonne 'Algorithme' indique quant à elle l'algorithme utilisé pour trouver cette solution.

Pour résoudre ces 13 cas, nous avons utilisé *IBBA* (algorithme 2) combiné avec la technique $rART_{\text{rAF2}}$ et la propagation des contraintes *CP* en limitant le temps de calcul à 1 heure et le nombre d'éléments stockés dans $\mathcal{L}$ à 2 millions. Dans le tableau 3.6, nous présentons les résultats de ces optimisations. Nous pouvons noter que :

- 3 problèmes tests de plus sont résolus, c'est-à-dire que l'optimalité de la solution disponible dans [81] a été prouvée, dont 2 de manière très efficace en seulement quelques minutes.
- Pour les 10 cas restants on peut remarquer que pour la plupart l'écart entre le majorant et le minorant reste petit, ce qui donne une bonne idée de la valeur du minimum global (voir ex6_1_1, ex6_1_3, ex6_2_10, ex6_2_13, ex9_2_6).

Ainsi, 3 problèmes tests supplémentaires ont été résolus. Grâce à notre technique de reformulation affine robuste *rART*, 64 cas sur les 74 ont été résolus.

Nom	N	M	ok ?	Minorant	UB de [81]	iter	t (s)	max $\lvert \mathcal{L} \rvert$
ex6_1_1	9	6	F	-0,020417460	-0,0202	1600000	3721,89	114516
ex6_1_3	13	9	F	-0,426770495	-0,3525	1000000	3892,91	241022
ex6_2_5	10	3	F	-7036,008545972	-70,7521	800000	4036,35	800000
ex6_2_7	10	3	F	-66,227346367	-0,1608	500000	4025,53	500000
ex6_2_10	7	3	F	-3,487775558	-3,0520	1200000	3769,27	697114
ex6_2_13	7	3	F	-0,275595143	-0,2162	1500000	3627,14	797896
ex7_2_3	8	6	**T**	7049,218001943	7049,2181	2031859	3328,74	56601
ex7_3_4	12	17	F	0,000000000	6,2746	900000	3610,82	200
ex7_3_5	13	15	F	-0,000001431	1,2036	500000	4259,42	291462
ex9_2_2	11	11	**T**	99,999499000	99,9995	589806	495,42	98305
ex9_2_6	17	12	F	-1,509887695	-1,0000	1200000	3819,71	705122
ex14_1_7	10	17	F	-10,664982242	0,0000	1100000	3708,7	8537
ex14_2_7	6	9	**T**	-0,000000010	0,0000	1	0,04	0

TABLE 3.6 – Résultats numériques de IBBA+*CP*+$rART_{\text{rAF2}}$ en donnant les meilleures solutions connues dès le début de l'algorithme.

3.6 Discussion

Dans ce chapitre, nous avons présenté un nouveau type de technique de relaxation linéaire fiable et robuste basée sur les formes affines et leurs arithmétiques. Ces méthodes construisent de manière automatique une relaxation linéaire de problèmes non convexes d'optimisation globale avec contraintes. Dans cette étude, nous avons intégré ces nouvelles techniques de relaxation affine dans notre propre algorithme de Branch and Bound par intervalles *IBBA* pour le calcul de minorants et l'élimination de sous-domaines ne contenant pas l'optimum global. L'efficacité de cette technique a été validée sur 74 problèmes tests issus de la librairie COCONUT. Le principal avantage de cette technique réside dans la façon dont sont générées les relaxations linéaires ayant le même nombre de variables que le problème original. En effet, très peu de temps est consacré à la résolution des programmes linéaires. De plus, lorsque la taille des sous-domaines étudiés devient petite, les erreurs générées par la relaxation sont réduites et les calculs de minorants deviennent très précis.

Ainsi, la combinaison de cette nouvelle technique de reformulation affine avec la technique de propagation des contraintes incluse dans un algorithme de Branch and Bound par intervalles permet d'obtenir un algorithme simple et efficace.

Chapitre 4

Algorithme de Branch and Bound par intervalles à mémoire limitée

Jusqu'à présent, pour résoudre de manière efficace les problèmes difficiles d'optimisation globale de type (PB), l'utilisateur devait toujours choisir entre des algorithmes de recherche locale, telle que la méthode du lagrangien augmenté, des algorithmes métaheuristiques basés sur une méthode de recherche locale, tels que les méthodes tabou [37] ou VNS [44], ou des techniques d'optimisation stochastique, telles que les algorithmes génétiques ou de recuit simulé.

$$\left\{\begin{array}{lll} \min\limits_{x\in\mathbf{X}\subset\mathbb{R}^n} & f(x) & \\ \text{s.t.} & g_i(x)\leq 0\ , & \forall i\in\{1,\ldots,p\},\\ & h_j(x)=0\ , & \forall j\in\{1,\ldots,q\}. \end{array}\right. \tag{PB}$$

Dans ce chapitre, nous proposons un nouveau type de métaheuristique basé sur un algorithme exact d'optimisation globale. Il reprend les éléments publiés dans [85]. Le principe consiste à accélérer la convergence de l'algorithme en limitant la mémoire nécessaire à son exécution. Ici, nous nous baserons sur notre code *IBBA* pour implémenter cette métaheuristique. Néanmoins, les résultats obtenus pourront très facilement être étendus à tous les autres algorithmes de Branch and Bound par intervalles, comme *GlobSol* [52].

L'idée de mettre une limite sur la mémoire disponible semble avoir été étudiée pour la première fois par Casado et al. dans [22]. Les auteurs y ont montré l'intérêt d'une telle méthode en l'appliquant à de nombreux exemples numériques de problèmes d'optimisation globale sans contrainte. Dans ce chapitre, cette méthodologie est présentée comme une approche métaheuristique. Nous nous intéresserons plus particulièrement à une heuristique permettant de conserver de l'information sur l'encadrement de la valeur du minimum global. De plus, des résultats théoriques sur la complexité en temps de ces algorithmes à mémoire limitée sont explicités. Notamment lorsque la structure de données est gérée en largeur d'abord, nous prouvons que la complexité en temps devient polynomiale ; la structure de données est ici nécessaire pour stocker les noeuds restant à visiter de l'arbre de Branch and Bound.

De nombreux tests numériques sur des problèmes continus d'optimisation globale avec contraintes viennent illustrer le comportement de l'heuristique donnant à la fin de son exécution une information sur l'encadrement de la valeur du minimum global.

Dans la section 4.1, nous présentons le principe métaheuristique basé sur la limitation de la mémoire. Puis, quelques heuristiques sont proposées, l'une d'entre elles, nommée *Hp*, y est détaillée ; *Hp* permet d'encadrer la valeur du minimum global en déterminant une précision certifiée P qui est calculée durant l'exécution de l'algorithme et ainsi de garantir l'optimalité P-globale de la solution, c'est-à-dire à une précision P près. Cette section contient aussi des propriétés sur l'heuristique *Hp*. L'objectif de cette heuristique n'est pas principalement de résoudre complètement le problème ou de trouver la meilleure solution, mais plutôt de savoir quelle est la plus petite précision pouvant être garantie de façon fiable et robuste avec la mémoire disponible. La section 4.2 est dédiée à l'étude de la complexité en temps d'une telle métaheuristique. Un des principaux résultats de ce travail concerne les théorèmes sur le calcul d'un majorant du nombre maximal d'itérations de la boucle principale de l'algorithme *IBBA*. Nous démontrons notamment que celui-ci est linéaire en fonction du nombre de variables, ce qui change l'ordre de complexité de l'algorithme en passant d'une complexité exponentielle à une complexité polynomiale. La section 4.3 valide notre approche métaheuristique en testant notre algorithme basé sur l'heuristique *Hp* sur des exemples numériques venant du site COCONUT [81]. Ces premiers résultats sont donnés uniquement pour illustrer le comportement de l'algorithme métaheuristique et montrer le potentiel de cette approche pour résoudre des problèmes d'optimisation globale beaucoup plus grands.

4.1 Principe métaheuristique

Pour clarifier l'étude de ces algorithmes métaheuristiques, nous rappellerons tout d'abord dans l'algorithme 10 le principe de *IBBA* utilisé dans ce chapitre. Cet algorithme possède exactement le même code que l'algorithme 2 décrit dans la section 1.2, à l'exception que les précisions ϵ_f et $\epsilon_{\mathcal{L}}$ sont maintenant des précisions absolues et non relatives.

L'algorithme 10 s'arrête en utilisant deux critères distincts :

(i) l'un sur la valeur de la fonction objectif telle que $\tilde{f} - \min_{(\mathbf{Z},f_z,R)\in\mathcal{L}} f_z \leq \epsilon_f$,

(ii) l'autre sur la largeur des boîtes qui sont stockées dans $\mathcal{L}$ telle que $\max_{(\mathbf{Z},f_z,R)\in\mathcal{L}} w(\mathbf{Z}) \leq \epsilon_{\mathcal{L}}$, où $w(\mathbf{Z})$ est la largeur de la boîte $\mathbf{Z}$.

De plus, nous devons ajouter un autre critère d'arrêt lorsque la structure de données est vide ; ce critère est non seulement nécessaire pour arrêter l'algorithme 10 si le problème n'a pas de solution réalisable, mais aussi lorsque nous inclurons notre algorithme à mémoire limitée. A la ligne 3 de l'algorithme 10, un élément est extrait de la structure de données suivant une heuristique : soit en largeur d'abord, soit le plus petit minorant d'abord, pour plus de détails voir [26]. Si une solution réalisable

existe, l'algorithme 10 retourne un point $\tilde{z}$ où la valeur du majorant $\tilde{f}$ du minimum global est atteint (tout dépend de la mémoire et du temps disponible pour résoudre le problème) ; de plus, une limite sur le temps de résolution ou sur la mémoire maximale utilisable peut être donnée par l'utilisateur et ajoutée aux critères d'arrêt.

Algorithme 10 Interval Branch and Bound Algorithm : IBBA

1: Soit $\mathbf{X}$ = le domaine initial dans lequel le minimum global est recherché,
$\tilde{f} = +\infty$, désigne le majorant courant du minimum global,
$R = [0, \ldots, 0]$, le vecteur de vérification des contraintes,
$\mathcal{L} = \{(\mathbf{X}, -\infty, R)\}$, l'initialisation de la structure de données pour stocker les éléments,
2: **répéter**
3: Extraire de $\mathcal{L}$ un élément $V = (\mathbf{Z}, f_z, R)$,
4: Division : $V_1 = (\mathbf{Z}_1, f_{z1}, R_1)$, $V_2 = (\mathbf{Z}_2, f_{z2}, R_2)$,
avec $R_1 = R_2 = R$, $f_{z1} = f_{z2} = f_z$, $\mathbf{Z} = \mathbf{Z}_1 \cap \mathbf{Z}_2$ et $\mathbf{Z}_1 \cup \mathbf{Z}_2 = \emptyset$,
5: **pour** $j = 1$ à 2 **faire**
6: Technique de réduction de $\mathbf{Z}_j$ par propagation des contraintes (voir section 1.3.2),
7: Mise à jour de f_{z_j} : un minorant de la fonction objectif sur $\mathbf{Z}_j$,
8: Vérification des contraintes et mise à jour de R_j,
9: **si** $\left(f_{z_j} \leq \tilde{f} - \epsilon_f\right)$ et aucune contrainte est insatisfaite **alors**
10: Insertion $(\mathbf{Z}_j, f_{z_j}, R_j)$ dans $\mathcal{L}$,
11: $\tilde{f} = \min\{\tilde{f}, f(mid(\mathbf{Z}_j))\}$, si et seulement si m satisfait toutes les contraintes,
12: **si** $\tilde{f}$ a changé **alors**
13: $\tilde{z} = mid(Z_j)$
14: Retirer de $\mathcal{L}$ tous les éléments $(\mathbf{Z}, f_z, R)$ tel que $\tilde{f} - \epsilon_f < f_z$
15: **finsi**
16: **finsi**
17: **fin pour**
18: *{insertion possible de l'algorithme 12}*
19: **jusqu'à** un critère d'arrêt {soit (i) ou (ii) ou les deux} ou lorsque $\mathcal{L} = \emptyset$

Notre métaheuristique consiste à fixer une borne supérieure `MaxElts` sur le nombre d'éléments que l'on peut stocker dans la structure de données $\mathcal{L}$. Lorsque cette borne est atteinte, une heuristique est utilisée pour limiter la taille de $\mathcal{L}$. Cette métaheuristique est complètement basée sur *IBBA* et, dans un premier temps, elle ne change pas le comportement de l'algorithme jusqu'à ce que la taille de $\mathcal{L}$ atteigne `MaxElts`. L'idée principale de cette métaheuristique est de permettre à *IBBA* de continuer la résolution du problème même lorsque toute la mémoire disponible est utilisée. L'heuristique est incluse dans l'algorithme 10, en ligne 18, mais l'on pourrait l'insérer à n'importe quel endroit de la boucle principale (entre la ligne 2 et 19 de l'algorithme 10).

Le but de cette heuristique est donc de choisir les éléments à éliminer de $\mathcal{L}$ pour

réduire sa taille. En suivant ce principe métaheuristique, de nombreuses heuristiques peuvent être proposées, gardant ou exploitant plus ou moins d'information sur le domaine que l'on étudie. De plus, contrairement aux métaheuristiques "classiques" telles que la recherche tabou [37] ou VNS [44], une étude complète du domaine est disponible lorsque l'heuristique est utilisée ; en effet, nous disposons d'un minorant de la fonction objectif sur chacun des sous-domaines stockés dans $\mathcal{L}$ et un vecteur indiquant quelles contraintes sont vérifiées sur le sous-domaine. Ainsi, toutes ces informations peuvent être utilisées par l'heuristique. Par exemple, différentes heuristiques simples peuvent être décrites comme suit :

- Insérer ligne 10 seulement un des deux éléments créés après l'étape de division ligne 4 de l'algorithme 10 (par exemple celui qui a le plus petit minorant f_z),
- Éliminer les éléments avec les plus grands minorants f_z,
- Éliminer les éléments dont les domaines ont les plus petites largeurs,
- Éliminer les éléments ayant la plus faible probabilité de satisfaire les contraintes, d'après un critère de satisfaction à déterminer,
- Éliminer les éléments ayant la plus faible probabilité de contenir le minimum global.

Beaucoup d'autres heuristiques peuvent être développées et testées suivant cette méthodologie métaheuristique.

Remarque 4.1.1 *En utilisant ce principe métaheuristique, on peut s'apercevoir que supprimer les éléments un par un de la structure de données $\mathcal{L}$ n'est pas nécessairement la meilleure solution, car elle peut nécessiter à chaque fois beaucoup de temps de calcul ou encore ne pas du tout accélérer la convergence de l'algorithme. Ainsi, il est conseillé de supprimer en une fois une partie plus importante de $\mathcal{L}$ lorsque sa taille atteint la limite* `MaxElts`*. Par exemple, il peut être intéressant d'éliminer 100, 1000 éléments ou encore* $10\% \times$ `MaxElts`*.*

L'heuristique, que nous avons choisi de développer dans cette partie, consiste à décaler la valeur de la solution courante $\tilde{f}$ pour garder dans $\mathcal{L}$ des éléments ayant les plus petits minorants ; nous noterons cette heuristique *Hp* (pour Heuristique sur la précision). Cette heuristique est détaillée dans l'algorithme 12.

Pour atteindre cet objectif, une nouvelle variable P est introduite. Cette variable représente la précision désirée sur la valeur du minimum global et elle va pouvoir être modifiée durant l'exécution de l'algorithme. A chaque fois que la taille de $\mathcal{L}$ atteint la limite `MaxElts`, P sera incrémentée. Les éléments de $\mathcal{L}$ dont le minorant f_z est supérieur à $\tilde{f} - P$ sont alors éliminés, comme on peut le voir en ligne 23 de l'algorithme 12. Inversement, lorsqu'une meilleure valeur de $\tilde{f}$ est trouvée, P peut diminuer. Ainsi, à la fin de l'exécution de l'algorithme, P nous donne une précision certifiée sur la solution $\tilde{f}$, i.e. la valeur du minimum global appartient avec certitude à l'intervalle $[\,\tilde{f} - P, \tilde{f}\,]$. L'algorithme 11 résume ce principe.

Remarque 4.1.2 *Les lignes 5 à 8, 11, 12, 17 à 19, 21, 24 et 26 à 29 de l'algorithme 12 ont été ajouté à l'algorithme 11 uniquement pour actualiser la valeur du pas d'incrémentation.*

Algorithme 11 Version simplifiée de l'heuristique sur la précision : LM_{Hp}

1: Initialiser la première fois P à ϵ_f. Notons `MaxElts`, la limite du nombre d'éléments que l'on peut stocker dans $\mathcal{L}$,
2: **si** $\tilde{f}$ a changé **alors**
3: Mettre à jour P
4: **finsi**
5: **si** $(\tilde{f} \neq \infty)$ et $(|\mathcal{L}| \geq$ `MaxElts`$)$ **alors**
6: Incrémenter P
7: Supprimer de $\mathcal{L}$ tous les éléments $(\mathbf{Z}, f_z, R)$ tels que $f_z > \tilde{f} - P$
8: **finsi**

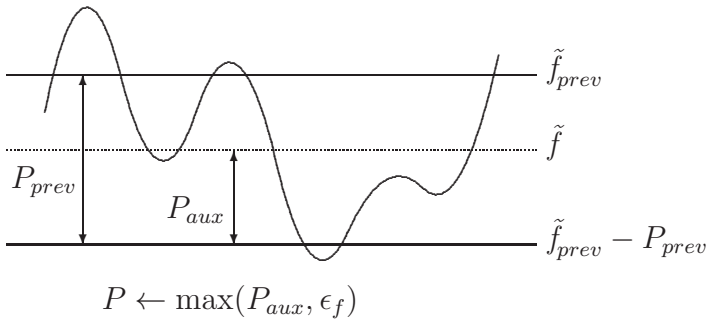


FIGURE 4.1 – Exemple de mise à jour de P (ligne 4 de l'algorithme 12)

A chaque itération de la boucle principale de l'algorithme 10, la sous-routine présentée dans l'algorithme 12 sera exécutée à la ligne 18 d'*IBBA* ; cet algorithme combinant l'algorithme 10 et 12 est noté *IBBA-LM*$_{Hp}$. Entre la ligne 2 et la ligne 14 de l'algorithme 12, lorsque $\tilde{f}$ a changé et si $\tilde{f} \in [\tilde{f}_{prev} - P, \tilde{f}_{prev}]$, la valeur de P est mise à jour de telle sorte que la borne $\tilde{f} - P$ ne soit pas modifiée ; i.e. $P = \tilde{f} - \tilde{f}_{prev} + P$; sinon P est réinitialisé à la valeur ϵ_f (voir la figure 4.1). Ainsi, si $\tilde{f}$ change, P décroît nécessairement. La variable ϵ_{aux} est utilisée pour représenter le pas d'incrémentation de P, voir ligne 22 ; en effet, comme nous l'avons mis en évidence dans la remarque 4.1.1, nous avons choisi d'incrémenter P de manière exponentielle pour ne pas perdre trop de temps dans l'algorithme 12. Ainsi, ϵ_{aux} est multiplié par 10 à la ligne 27 lorsque l'algorithme *Hp* est utilisé 10 fois ; inversement, si $\tilde{f}$ a changé et donc que P décroît, ϵ_{aux} est actualisé pour tenir compte de la nouvelle valeur de P ligne 6-8 ou ligne 11. A la ligne 17, si tous les éléments de $\mathcal{L}$ peuvent être éliminés en une incrémentation de P, i.e. $\tilde{f} - (P + \epsilon_{aux}) \leq \min_{(\mathbf{Z}, f_z, R) \in \mathcal{L}} f_z$, ϵ_{aux} est divisé par 10 jusqu'à ce que des éléments de $\mathcal{L}$ soient conservés, i.e. $\tilde{f} - (P + \epsilon_{aux}) > \min_{(\mathbf{Z}, f_z, R) \in \mathcal{L}} f_z$. Ainsi, durant l'exécution de l'algorithme *IBBA-LM*$_{Hp}$, P peut croître si `MaxElts` est atteint et décroître si une meilleure valeur de $\tilde{f}$ est trouvée.

Remarque 4.1.3 *Pour* IBBA-LM$_{Hp}$*, il n'est pas nécessaire de supprimer à chaque fois une grande partie de* $\mathcal{L}$*, car l'élimination s'effectue de manière indirecte par l'accroissement de la valeur* P *; en effet, plus* P *sera grand et plus il y aura d'éléments supprimés.*

Algorithme 12 Heuristique sur la précision : LM_{Hp}

Initialisation avant la première utilisation de l'algorithme :
Soit $P = \epsilon_f$, $\epsilon_{aux} = \epsilon_f$ et $i = 1$. Notons `MaxElts`, désigne la limite du nombre d'éléments que l'on peut stocker dans $\mathcal{L}$ et $\tilde{f}_{prev}$, la valeur précédente du minimum courant $\tilde{f}$,
si $\tilde{f}$ a changé **alors**
  si $(\tilde{f} \neq \infty)$ et $(\tilde{f} \geq \tilde{f}_{prev} - P)$ **alors**
    $P = \max\{\tilde{f} - \tilde{f}_{prev} + P, \epsilon_f\}$
    $\epsilon_{aux} = \epsilon_f$
    tantque $\epsilon_{aux} \leq \frac{P}{10 \times i}$ **faire**
      $\epsilon_{aux} = 10 \times \epsilon_{aux}$
    fin tantque
  sinon
    $P = \epsilon_f$
    $\epsilon_{aux} = \epsilon_f$
    $i = 1$
  finsi
finsi
si $(\tilde{f} \neq \infty)$ et $(|\mathcal{L}| \geq$ `MaxElts`$)$ **alors**
  si $\tilde{f} - P > \min_{(\mathbf{Z}, f_z, R) \in \mathcal{L}} f_z$ **alors**
    tantque $\tilde{f} - P - \epsilon_{aux} < \min_{(\mathbf{Z}, f_z, R) \in \mathcal{L}} f_z$ **faire**
      $\epsilon_{aux} = \frac{\epsilon_{aux}}{10}$
    fin tantque
  finsi
  $\epsilon_{aux} = \max\{\epsilon_{aux}, \epsilon_f\}$
  $P = P + \epsilon_{aux}$
  Supprimer de $\mathcal{L}$ tous les éléments $(\mathbf{Z}, f_z, R)$ tels que $f_z > \tilde{f} - P$
  $i = i + 1$
finsi
si $i = 10$ **alors**
  $\epsilon_{aux} = 10 \times \epsilon_{aux}$
  $i = 1$
finsi

Proposition 4.1.4 *A la fin de l'algorithme* IBBA-LM$_{Hp}$ *muni du critère d'arrêt* $\tilde{f} - \min_{(\mathbf{Z},f_z,R)\in\mathcal{L}} f_z \leq \epsilon_f$*, si une solution réalisable existe, alors la valeur du minimum global appartient à l'intervalle* $[\,\tilde{f}-P,\tilde{f}\,]$.

Démonstration. Si le nombre d'éléments de $\mathcal{L}$ n'atteint jamais `MaxElts`, aucune heuristique incluse dans l'algorithme 10 n'est utilisée et $P = \epsilon_f$ (sa valeur initiale). Ainsi, l'algorithme 10 est exécuté tout seul et fournit un encadrement certifié de la valeur du minimum global avec une largeur inférieure à $P = \epsilon_f$.
Dans le cas contraire, l'algorithme 12 est utilisé et supprime donc de $\mathcal{L}$ les boîtes $(\mathbf{Z}, f_z, R)$ telles que $f_z > \tilde{f} - P$. Montrons que $\tilde{f} - P$ est bien un minorant certifié de la valeur du minimum global. Si P croît (ligne 22 de l'algorithme 12), un élément de $\mathcal{L}$ est supprimé si et seulement si il est prouvé qu'il ne peut contenir une solution réalisable inférieure à $\tilde{f} - P$. Notons P_{prev} la valeur de P à l'itération précédente (P_{prev} correspond à la valeur de P avant la ligne 4 de l'algorithme 12). Si $\tilde{f}$ a changé, P décroît de telle sorte que $P = \max\{\tilde{f} - \tilde{f}_{prev} + P_{prev}, \epsilon_f\}$ (ligne 4 de l'algorithme 12), ainsi tous les éléments déjà supprimés ne peuvent pas contenir une solution inférieure à $\tilde{f} - P$ car $\tilde{f} - P \leq \tilde{f}_{prev} - P_{prev}$. Ainsi, à la fin de l'algorithme, les éléments supprimés sont ceux qui ne satisfont pas une contrainte ou ne contiennent pas de solution réalisable inférieure à $\tilde{f} - P$. C'est pourquoi, à la fin de l'exécution de *IBBA-LM*$_{Hp}$, $\tilde{f} - P$ est un minorant certifié de la valeur du minimum global. De plus, $\tilde{f}$ correspond à la meilleure solution trouvée, ce qui implique qu'il soit également un majorant de la valeur du minimum global.
Pour conclure, l'intervalle $[\,\tilde{f} - P, \tilde{f}\,]$ est donc bien un encadrement certifié de la valeur du minimum global. ∎

Remarque 4.1.5 *Afin d'utiliser* IBBA-LM$_{Hp}$ *de manière efficace, il peut être intéressant d'effectuer une recherche de minima locaux avant son exécution. En effet, ce minimum pourrait être utilisé dans l'algorithme afin d'initialiser la variable* $\tilde{f}$*. Car, dans le cas où aucune solution réalisable n'a encore été trouvée, il est tout de même possible que la taille de* $\mathcal{L}$ *atteigne la limite* `MaxElts` *avec* $\tilde{f} = +\infty$*, ce qui rend inefficace notre heuristique Hp. Une autre heuristique devra alors être utilisée au préalable.*

4.2 Étude de la complexité des algorithmes *IBBA-LM*

Dans cette partie, nous étudierons la complexité en temps de la méthode métaheuristique *IBBA-LM* (algorithme 10 et 12) décrite dans la partie 4.1. Nous montrerons également que lorsque la structure de données $\mathcal{L}$ est gérée en largeur d'abord, la complexité de ces algorithmes basés sur cette méthode métaheuristique à mémoire limitée change, i.e. la complexité devient linéaire en fonction du nombre d'itérations de la boucle principale de l'algorithme 10 (de la ligne 2 à 19).

Par la suite, nous noterons $IBBA^{\texttt{bf}}$ et $IBBA\text{-}LM^{\texttt{bf}}$, les algorithmes *IBBA* et *IBBA-LM* ayant leur structure de données $\mathcal{L}$ gérée en largeur d'abord, i.e. les premiers éléments sont ceux ayant la plus grande largeur $w(\mathbf{Z})$ (cet ordre est choisi, car il ne dépend pas du calcul des minorants f_z). Les notations $IBBA_{\epsilon_f}(\mathbf{X})$ et $IBBA_{\epsilon_{\mathcal{L}}}(\mathbf{X})$ sont aussi utilisées pour indiquer que l'algorithme *IBBA* est initialisé avec $\mathbf{X}$ comme domaine de recherche initial et utilise comme critère d'arrêt respectivement : (i) celui sur la précision ϵ_f obtenu sur l'encadrement de la valeur du minimum global ; (ii) celui sur la largeur maximale $\epsilon_{\mathcal{L}}$ des éléments encore stockés dans $\mathcal{L}$.

Notons $P_{IBBA}(n)$ la complexité des sous-fonctions à l'intérieur de la boucle principale de l'algorithme *IBBA* pour une instance du problème (PB) avec n variables. Cette complexité $P_{IBBA}(n)$ dépend bien entendu des procédures d'accélération qui sont utilisées dans le code *IBBA*, comme la technique de propagation décrite dans la section 1.3.2, ou un type de technique utilisé pour calculer les minorants (voir chapitre 2), etc. Néanmoins, cette complexité est, dans le pire des cas, polynomiale en temps et les procédures n'utilisent généralement qu'une petite partie de la mémoire.

Rappelons la définition classique de la largeur d'un intervalle $Z \in \mathbb{I}^n$: $w(\mathbf{Z}) = \max_{i\in\{1,\ldots,n\}} w(\mathbf{z}_i) = \max_{i\in\{1,\ldots,n\}}(\overline{\mathbf{z}_i} - \underline{\mathbf{z}_i})$, où $\mathbf{z}_i = [\underline{\mathbf{z}_i}, \overline{\mathbf{z}_i}]$ est le i$^{\text{ème}}$ composant de $\mathbf{Z}$, et où $w(\mathbf{z}_i) = (\overline{\mathbf{z}_i} - \underline{\mathbf{z}_i})$ est la largeur du i$^{\text{ème}}$ composant $\mathbf{z}_i$ de $\mathbf{Z}$.

Tout d'abord, nous nous sommes intéressés à des propriétés particulières de l'arbre binaire complet, noté $\mathfrak{B}(\mathbf{X})$, qui est créé implicitement par les bissections successives des éléments par le milieu de leur plus large composante. Cet arbre binaire $\mathfrak{B}(\mathbf{X})$ est construit jusqu'à ce que la largeur de tous les sous-domaines devienne plus petite que $\epsilon_{\mathcal{L}}$. Par la suite, nous utiliserons la définition classique du niveau d'un arbre binaire ; le niveau 0 se situe à la racine $\mathbf{X}$, au niveau 1, nous avons uniquement les deux sous-domaines générés à partir de $\mathbf{X}$ en divisant en deux sa plus large composante, etc. A chaque niveau, nous obtenons 2^k sous-domaines générés par k bissections successives.

Proposition 4.2.1 *Les largeurs de toutes les composantes de chacun des sous-domaines d'un même niveau de l'arbre $\mathfrak{B}(\boldsymbol{X})$ sont égales.*

Démonstration. L'idée de cette démonstration est de prouver que si deux sous-domaines du même niveau de $\mathfrak{B}(\mathbf{X})$ ont la largeur d'au moins une composante qui diffère, les pères de ces deux éléments diffèrent aussi de la largeur d'au moins un élément. Ainsi, par récursivité jusqu'à la racine, on obtient une contradiction.

Tout d'abord, notons que deux sous-domaines issus de la bissection du même domaine ont exactement les mêmes composantes excepté une, ayant la même largeur (due à la bissection) ; ces deux sous-domaines sont sur le même niveau de $\mathfrak{B}(\mathbf{X})$.

Supposons maintenant qu'il existe deux sous-domaines $\mathbf{Z}_1$ et $\mathbf{Z}_2$ sur le même niveau k de $\mathfrak{B}(\mathbf{X})$ dont la largeur d'au moins une composante diffère ; notons la coordonnée correspondante par ν. Ceci implique d'après la remarque du début de la démonstration que $\mathbf{Z}_1$ et $\mathbf{Z}_2$ sont issus de la décomposition de deux domaines distincts du niveau $k-1$, c'est-à-dire qu'ils n'ont pas le même père ; notons ces deux pères $\mathbf{W}_1$ et $\mathbf{W}_2$ respectivement. Au moins une largeur d'une composante de $\mathbf{W}_1$ diffère de celle de $\mathbf{W}_2$ par l'une des 4 manières suivantes :

- $\mathbf{W}_1$ et $\mathbf{W}_2$ ont la même $\nu^{\text{ème}}$ composante que $\mathbf{Z}_1$ et respectivement $\mathbf{Z}_2$, ainsi $\mathbf{W}_1$ et $\mathbf{W}_2$ diffèrent d'au moins un élément ;
- $\mathbf{W}_1$ a la même composante ν que $\mathbf{Z}_1$, mais pas $\mathbf{W}_2$, i.e. la largeur de la composante ν de $\mathbf{Z}_2$ est multipliée par deux dans $\mathbf{W}_2$. Ceci implique que dans $\mathbf{W}_1$ une autre composante $\mu \neq \nu$ a été choisie pour réaliser la bissection et ainsi les largeurs des composantes μ de $\mathbf{W}_1$ et $\mathbf{W}_2$ diffèrent ;
- $\mathbf{W}_2$ a la même composante ν, mais pas $\mathbf{W}_1$, par un raisonnement analogue au cas précédent, on peut conclure qu'il existe deux composantes de $\mathbf{W}_1$ et $\mathbf{W}_2$ qui ont des largeurs différentes ;
- $\mathbf{W}_1$ et $\mathbf{W}_2$ n'ont pas les mêmes composantes ν que $\mathbf{Z}_1$ et respectivement $\mathbf{Z}_2$, cela veut dire que la largeur de la composante ν de $\mathbf{W}_1$ et $\mathbf{W}_2$ vaut deux fois celle de $\mathbf{Z}_1$ et respectivement $\mathbf{Z}_2$, donc elles diffèrent.

Ainsi, par récursivité, on obtient que les deux premiers sous-domaines du niveau 1 de $\mathfrak{B}(\mathbf{X})$ ont la largeur d'au moins une composante qui diffère, ce qui est une contradiction vu qu'ils sont issus de la bissection du domaine situé à la racine. ∎

Corollaire 4.2.2 *Tous les sous-domaines d'un même niveau de $\mathfrak{B}(\boldsymbol{X})$ ont exactement la même largeur.*

Démonstration. Ce corollaire est juste une conséquence de la proposition 4.2.1. ∎

4.2.1 Critère d'arrêt sur la largeur des sous-domaines

Dans cette sous-section, nous n'allons considérer que le critère d'arrêt sur la largeur des domaines :

$$\max_{(\mathbf{Z},f_z,R)\in\mathcal{L}} w(\mathbf{Z}) \leq \epsilon_{\mathcal{L}}, \tag{4.1}$$

avec $\epsilon_{\mathcal{L}}$ une petite valeur positive donnée par l'utilisateur de l'algorithme 10. Nous noterons $IBBA_{\epsilon_{\mathcal{L}}}$ l'algorithme utilisant ce critère d'arrêt, voir équation (4.1).

A chaque itération de la boucle principale de $IBBA_{\epsilon_{\mathcal{L}}}$ avec ou sans limitation de la mémoire, un élément est divisé en deux sous-domaines. Nous avons donc 3 cas :

- les 2 sous-domaines sont réinsérés dans $\mathcal{L}$,
- seul l'un des 2 éléments créés est inséré dans $\mathcal{L}$,
- aucun des 2 n'est inséré.

Le domaine initial ayant produit ces deux éléments a été extrait de $\mathcal{L}$, ainsi dans le pire des cas, le nombre d'éléments stockés dans $\mathcal{L}$ croît au plus d'un à chaque itération de la boucle principale de l'algorithme $IBBA_{\epsilon_{\mathcal{L}}}$. Donc dans le pire des cas et sans utiliser la limitation de la mémoire, le nombre d'éléments stockés dans $\mathcal{L}$ est égal au nombre d'itérations de la boucle principale de $IBBA_{\epsilon_{\mathcal{L}}}$ plus un.

Proposition 4.2.3 *En notant $\mathcal{N}(\mathrm{IBBA}_{\epsilon_{\mathcal{L}}}(\boldsymbol{X}))$ le nombre maximal d'itérations de $\mathrm{IBBA}_{\epsilon_{\mathcal{L}}}$ pour résoudre un problème du type (PB) sur le domaine initial $\boldsymbol{X}$, on a :*

$$\mathcal{N}(\mathrm{IBBA}_{\epsilon_{\mathcal{L}}}(\boldsymbol{X})) = 2^{\sum_{i=1}^{n} \left\lceil \log_2 \frac{w(\mathbf{x}_i)}{\epsilon_{\mathcal{L}}} \right\rceil} - 1, \tag{4.2}$$

et un majorant est :

$$\mathcal{N}(\text{IBBA}_{\epsilon_{\mathcal{L}}}(\boldsymbol{X})) \leq 2^{n \times \left\lceil \log_2 \frac{w(\boldsymbol{X})}{\epsilon_{\mathcal{L}}} \right\rceil}. \tag{4.3}$$

Démonstration. En remarquant que le nombre maximal d'itérations de la boucle principale de $IBBA_{\epsilon_{\mathcal{L}}}$ utilisé pour résoudre le problème sur le domaine initial $\mathbf{X}$ est obtenu lorsque tout l'arbre $\mathfrak{B}(\mathbf{X})$ est construit et exploré, il suffit juste de compter le nombre d'itérations pour construire l'arbre en entier. Pour avoir tous les sous-domaines $\mathbf{Z}$ avec une largeur $w(\mathbf{Z})$ plus petite que $\epsilon_{\mathcal{L}}$, il est nécessaire de bissecter K_i fois sur chaque composante i jusqu'à ce que $\frac{w(\mathbf{z}_i)}{2^{K_i}} \leq \epsilon_{\mathcal{L}}$. Ainsi $K_i = \left\lceil \log_2 \frac{w(\mathbf{z}_i)}{\epsilon_{\mathcal{L}}} \right\rceil$. En partant de la décomposition de chaque composante, on génère ainsi $\prod_{i=1}^{n} 2^{K_i}$ sous-domaines de largeur inférieure à $\epsilon_{\mathcal{L}}$. En utilisant le corollaire 4.2.2, on sait qu'à chaque niveau de $\mathfrak{B}(\mathbf{X})$ tous les sous-domaines sont de largeur égale, donc au dernier niveau de $\mathfrak{B}(\mathbf{X})$ nous avons $\prod_{i=1}^{n} 2^{K_i} = 2^{\sum_{i=1}^{n} K_i}$ sous-domaines de largeur au plus $\epsilon_{\mathcal{L}}$.

En notant le niveau 0 comme étant le premier niveau de $\mathfrak{B}(\mathbf{X})$, on obtient un lien entre le nombre de niveaux et le nombre d'éléments par niveau ; au niveau k on a 2^k éléments. Ainsi, le dernier niveau de $\mathfrak{B}(\mathbf{X})$ est le niveau $\sum_{i=1}^{n} K_i$. Pour compter le nombre de bissections de $\mathfrak{B}(\mathbf{X})$ correspondant au nombre maximal d'itérations de la boucle principale de $IBBA_{\epsilon_{\mathcal{L}}}$, nous n'avons plus qu'à additionner tous les éléments de chaque niveau de $\mathfrak{B}(\mathbf{X})$ jusqu'au dernier niveau :

$$\mathcal{N}(IBBA_{\epsilon_{\mathcal{L}}}(\mathbf{X})) = \sum_{l=0}^{\sum_{i=1}^{n} K_i - 1} 2^l = 2^{\sum_{i=1}^{n} K_i} - 1,$$

Pour le majorant, il est directement obtenu lorsque l'inégalité $w(\mathbf{z}_i) \leq w(\mathbf{Z})$, $\forall i \in \{1, \cdots, n\}$ est utilisée. ∎

Corollaire 4.2.4 *La complexité en temps de l'algorithme* $\text{IBBA}_{\epsilon_{\mathcal{L}}}$ *est de l'ordre de grandeur :*

$$\mathcal{O}\left(\text{IBBA}_{\epsilon_{\mathcal{L}}}(\boldsymbol{X})\right) = 2^n. \tag{4.4}$$

Pour être plus précis, on doit multiplier la complexité par $P_{IBBA_{\epsilon_{\mathcal{L}}}}(n)$ *(correspondant à la complexité des procédures d'accélération incluses), cependant leur ordre est polynomial.*

Démonstration. Ce résultat est directement obtenu à partir de l'équation (4.3) de la proposition 4.2.3. ∎

Remarque 4.2.5 *Notons que la complexité est la même lorsque des heuristiques différentes sont utilisées pour la gestion de la structure de données* $\mathcal{L}$ *: en profondeur d'abord, en largeur d'abord, par ordre croissant des minorants* f_z*, etc.*

Les résultats théoriques précédents sont bien connus et la plupart part d'entre eux peuvent être trouvés également dans [25] ; ici, nous les avons rappelés et adaptés à nos

notations et à notre algorithme *IBBA*. Les propriétés suivantes sont, quant à elles, nouvelles et traitent de la complexité induite par l'utilisation de la métaheuristique que nous proposons dans ce chapitre.

Maintenant, nous allons donc nous intéresser à la complexité de notre métaheuristique nommée *IBBA-LM*. Les résultats suivants sur la complexité en temps ne sont valables que pour *IBBA-LM*$^{\texttt{bf}}$ (avec la structure de données triée en largeur d'abord), qui est un cas particulier de l'algorithme *IBBA-LM*.

Notons $\mathcal{N}(IBBA(\mathbf{X}))$ le nombre maximal d'itérations de la boucle principale de *IBBA* (algorithme 10) incluant ou non la technique de limitation de la mémoire.

Proposition 4.2.6 *Si* $\mathcal{N}(\text{IBBA}_{\epsilon_\mathcal{L}}(\boldsymbol{X})) \leq \texttt{MaxElts}$ *alors*

$$\mathcal{N}(\text{IBBA-LM}_{\varepsilon_\mathcal{L}}(\boldsymbol{X})) = \mathcal{N}(\text{IBBA}_{\epsilon_\mathcal{L}}(\boldsymbol{X})) = 2^{\sum_{i=1}^{n} \left\lceil \log_2 \frac{w(\mathbf{x}_i)}{\epsilon_\mathcal{L}} \right\rceil} - 1,$$

i.e., si le nombre maximal d'itérations de $\text{IBBA}_{\epsilon_\mathcal{L}}$ *est inférieure à* `MaxElts`, $\text{IBBA}_{\epsilon_\mathcal{L}}$ *et* $\text{IBBA-LM}_{\varepsilon_\mathcal{L}}$ *ont le même nombre d'itérations.*

Démonstration. Ceci est évident, car si la limite de la mémoire n'est pas atteinte alors l'heuristique de *IBBA-LM*$_{\varepsilon_\mathcal{L}}$ n'est jamais utilisée et les nombres d'itérations de *IBBA-LM*$_{\varepsilon_\mathcal{L}}$ et de *IBBA*$_{\epsilon_\mathcal{L}}$ sont exactement les mêmes. Donc, dans le pire des cas (générant le nombre maximal d'itérations), le résultat est prouvé. ∎

Ainsi, si `MaxElts` est trop grand, notre métaheuristique ne sera jamais utilisée. Néanmoins, cette métaheuristique est utilisée en pratique, car elle empêche l'algorithme de dépasser la capacité de la mémoire vive de l'ordinateur.

Considérons maintenant le cas intéressant c'est-à-dire lorsque l'algorithme *IBBA* nécessite plus de mémoire que celle disponible physiquement sur l'ordinateur. Dans ces hypothèses, la limitation de la mémoire est utilisée et va produire quelques changements dans le comportement de *IBBA*.

Théorème 4.2.7 *Un majorant du nombre maximal d'itérations de la boucle principale d'un algorithme métaheuristique* $\text{IBBA-LM}^{\texttt{bf}}_{\epsilon_\mathcal{L}}$ *est donné par :*

$$\mathcal{N}(\text{IBBA-LM}^{\texttt{bf}}_{\epsilon_\mathcal{L}}(\boldsymbol{X})) \leq \texttt{MaxElts} \times \left(\sum_{i=1}^{n} \left\lceil \log_2 \frac{w(\mathbf{x}_i)}{\epsilon_\mathcal{L}} \right\rceil \right).$$

Démonstration. Le nombre maximal de sous-domaines considérés est borné par le nombre maximal de domaines par niveau de l'arbre $\mathfrak{B}(\mathbf{X})$ étant stocké dans $\mathcal{L}$ (c'est-à-dire `MaxElts`) multiplié par le nombre maximal de niveaux de $\mathfrak{B}(\mathbf{X})$. ∎

Pour être plus précis nous avons :

$$\mathcal{N}(IBBA\text{-}LM^{\texttt{bf}}_{\epsilon_\mathcal{L}}(\mathbf{X})) \leq \min \left\{ \mathcal{N}(IBBA_{\epsilon_\mathcal{L}}(\mathbf{X})), \texttt{MaxElts} \times \left(\sum_{i=1}^{n} \left\lceil \log_2 \frac{w(\mathbf{x}_i)}{\epsilon_\mathcal{L}} \right\rceil \right) \right\}.$$

Néanmoins, nous n'étudierons ici que la complexité en temps lorsque la limitation de la mémoire a un effet sur le code *IBBA*.

Théorème 4.2.8 *L'ordre de complexité en temps d'un algorithme métaheuristique* IBBA-LM$^{\mathtt{bf}}_{\epsilon_{\mathcal{L}}}$ *est :*

$$\mathcal{O}(\text{IBBA-LM}^{\mathtt{bf}}_{\epsilon_{\mathcal{L}}}(\boldsymbol{X})) = n \times P_{IBBA\text{-}LM_{\epsilon_{\mathcal{L}}}}(n).$$

Afin d'être plus précis l'ordre de complexité doit être multiplié par une constante qui peut prendre une valeur très grande : $\mathtt{MaxElts} \times \left\lceil \log_2 \frac{w(\boldsymbol{X})}{\epsilon_{\mathcal{L}}} \right\rceil$.

Démonstration. En utilisant le théorème 4.2.7, on peut obtenir un autre majorant en considérant le calcul suivant : $\forall i \in \{1, \cdots, n\}, w(\mathbf{x}_i) \leq w(\mathbf{X})$. Ainsi :

$$\mathtt{MaxElts} \times \left(\sum_{i=1}^{n} \left\lceil \log_2 \frac{w(\mathbf{x}_i)}{\epsilon_{\mathcal{L}}} \right\rceil \right) \leq n \times \mathtt{MaxElts} \times \left\lceil \log_2 \frac{w(\mathbf{X})}{\epsilon_{\mathcal{L}}} \right\rceil,$$

et le résultat précédent découle en incluant la complexité polynomiale $P_{IBBA\text{-}LM_{\epsilon_{\mathcal{L}}}}(n)$ qui est intrinsèque aux techniques d'accélération incluses dans $IBBA\text{-}LM^{\mathtt{bf}}_{\epsilon_{\mathcal{L}}}$. ∎

Considérons $\mathcal{A}_{\max}$ un algorithme $IBBA\text{-}LM^{\mathtt{bf}}_{\epsilon_{\mathcal{L}}}$ qui atteint le nombre maximal d'itérations. Notons $\mathcal{L}$ une structure de données triée en largeur d'abord pour stocker les sous-domaines étudiés, notons alors $\mathcal{L}(k)$ l'état de cette structure de données après k itérations de l'algorithme. Si la solution est obtenue en moins de k itérations, on prendra par convention que $\mathcal{L}(k) = \emptyset$ et $\max_{(\mathbf{Z}, f_z, R) \in \mathcal{L}(k)} w(\mathbf{Z}) = 0$.

Notons $\mathcal{L}_{\max}$ la structure de données de $\mathcal{A}_{\max}$. Dans $\mathcal{A}_{\max}$, à chaque itération de la boucle principale, aucune élimination ni réduction n'est effectuée (la décomposition est simplement effectuée en choisissant l'élément le plus large et en découpant en deux au milieu de sa plus large composante). Ainsi, la taille de $\mathcal{L}_{\max}$ va rapidement croître jusqu'à atteindre la limite `MaxElts`. Par la suite, nous allons montrer que le comportement de cet algorithme génère le plus grand nombre possible d'itérations pour un algorithme de type $IBBA\text{-}LM^{\mathtt{bf}}_{\epsilon_{\mathcal{L}}}$.

Lemme 4.2.9 *Notons* $\mathcal{L}$ *une structure de données utilisée durant l'exécution d'un algorithme* IBBA-LM$^{\mathtt{bf}}_{\epsilon_{\mathcal{L}}}$. *On obtient alors :*

$$\max_{(\boldsymbol{Z}, f_z, R) \in \mathcal{L}(k)} w(\boldsymbol{Z}) \leq \max_{(\boldsymbol{Z}, f_z, R) \in \mathcal{L}_{\max}(k)} w(\boldsymbol{Z}), \forall k \in \{1, \cdots, \mathcal{N}(\text{IBBA}_{\epsilon_{\mathcal{L}}}(\boldsymbol{X}))\}.$$

Démonstration. Tout d'abord, notons qu'à l'itération $\mathtt{MaxElts}-1$, `MaxElts` éléments sont stockés dans $\mathcal{L}_{\max}$ c'est-à-dire le maximum autorisé (tous les sous-domaines sont gardés). Ainsi, de l'itération 1 à $\mathtt{MaxElts}-1$, $\max_{(\mathbf{Z}, f_z, R) \in \mathcal{L}(k)} w(\mathbf{Z}) \leq \max_{(\mathbf{Z}, f_z, R) \in \mathcal{L}_{\max}(k)} w(\mathbf{Z})$.

Par la suite, $l_{\inf}(\mathcal{A}, k)$ désigne le niveau le plus haut dans $\mathfrak{B}(\mathbf{X})$ encore stocké pour un algorithme $\mathcal{A}$ à l'itération k. De plus, notons $n_{\max}$ le nombre maximal d'éléments du niveau l de $\mathfrak{B}(\mathbf{X})$ encore stocké dans $\mathcal{L}_{\max}$ avec $l = l_{\inf}(\mathcal{A}_{\max}, \mathtt{MaxElts}-1)$.

A l'itération $\mathtt{MaxElts}-1+n_{\max}$, on a dans le pire des cas `MaxElts` éléments dans $\mathcal{L}_{\max}$; la limite est atteinte depuis l'itération $\mathtt{MaxElts}-1$. Ainsi, $l_{\inf}(\mathcal{A}_{\max}, \mathtt{MaxElts}-1+n_{\max}) = l+1$. En effet, entre l'itération $\mathtt{MaxElts}-1$ et l'itération $\mathtt{MaxElts}-1+$

$n_{\max}$, les $n_{\max}$ éléments extraits de $\mathcal{L}_{\max}$ sont les $n_{\max}$ éléments du niveau l de $\mathfrak{B}(\mathbf{X})$ ($\mathcal{L}$ étant triée en largeur d'abord). Ainsi, pour ne pas dépasser la limite `MaxElts` et enclencher l'heuristique qui réduirait le nombre d'éléments stockés et le nombre d'itérations, nous ne pouvons donc insérer qu'un seul des deux sous-domaines générés à chaque itération. Donc, $l_{\inf}(\mathcal{A}_{\max}, \texttt{MaxElts} - 1 + n_{\max}) = l + 1$.

De plus, on a $l_{\inf}(\mathcal{A}, \texttt{MaxElts} - 1 + n_{\max}) \geq l + 1$ (avec $\mathcal{A}$ correspondant à $\mathcal{L}$), car si $l_{\inf}(\mathcal{A}, \texttt{MaxElts} - 1) = l$, cela implique que le nombre d'éléments dans $\mathcal{L}$, de niveau l et à l'itération $\texttt{MaxElts} - 1$, est inférieur à $n_{\max}$, ce qui est impossible d'après la construction de $\mathcal{L}_{\max}$, dans laquelle chaque boîte générée est gardée jusqu'à l'itération $\texttt{MaxElts} - 1$. Après $k \times \texttt{MaxElts}$ itérations de plus, on a $l_{\inf}(\mathcal{A}_{\max}, \texttt{MaxElts} - 1 + n_{\max} + k \times \texttt{MaxElts}) = l + 1 + k$. On note qu'il est impossible de descendre plus bas dans les niveaux de $\mathfrak{B}(\mathbf{X})$ sans insérer plus de `MaxElts` éléments d'un même niveau de $\mathfrak{B}(\mathbf{X})$.
Pour conclure, il est impossible d'avoir à l'itération $k > \texttt{MaxElts} - 1$, la relation suivante $l_{\inf}(\mathcal{A}_{\max}, k) > l_{\inf}(\mathcal{A}, k)$. Ceci implique, d'après la proposition 4.2.1, qu'il y a un cas où il existe un sous-domaine $\mathbf{W} \in \mathcal{L}(k)$ tel que $w(\mathbf{W}) > \max_{(\mathbf{Z}, f_z, R) \in \mathcal{L}_{\max}(k)} w(\mathbf{Z})$.
Ainsi, le lemme est démontré. ∎

Remarque 4.2.10 *Le résultat du lemme 4.2.9 est toujours vrai pour tous les algorithmes* IBBA-LM$^{\texttt{bf}}_{\epsilon_{\mathcal{L}}}$ *incluant ou non des techniques d'accélération. Ceci est dû au fait que l'algorithme* $\mathcal{A}_{\max}$ *génère toujours le nombre maximal d'itérations possible. De plus, ces propriétés sont toujours vraies si l'on considère aussi les algorithmes de Branch and Bound par intervalles pour résoudre des problèmes d'optimisation globale sans contrainte, tel que celui décrit dans [22] ; il est seulement nécessaire d'avoir une technique de division par le milieu de la plus large composante et que les données soient triées en largeur d'abord dans* $\mathcal{L}$*. Ainsi, ces résultats théoriques peuvent expliquer partiellement pourquoi l'algorithme par intervalles présenté dans [22] est si efficace.*

Pour l'instant, nous ne pouvons rien dire si la structure de données $\mathcal{L}$ est gérée autrement, par exemple en profondeur, ou en considérant l'élément qui a le plus petit minorant d'abord.

4.2.2 Critère d'arrêt sur la précision obtenue de la valeur du minimum global

Généralement dans *IBBA*, nous utilisons le test (4.5) pour stopper l'exécution du code lorsque l'on veut encadrer précisément la valeur du minimum global :

$$\tilde{f} - \min_{(\mathbf{Z}, f_z, R) \in \mathcal{L}} f_z \leq \epsilon_f, \tag{4.5}$$

avec $\tilde{f}$ la meilleure solution réalisable trouvée durant l'exécution de *IBBA*, voir l'algorithme 10.

Afin d'obtenir un résultat similaire concernant la complexité en temps, nous devons supposer que la valeur du minimum global, noté f^*, a été trouvée, i.e. $f^* = \tilde{f}$. En utilisant la définition de l'α-convergence des fonctions d'inclusion utilisées pour calculer les encadrements de f (définition 1.1.13), on a l'inégalité suivante :

$$\forall(\mathbf{Z}, f_z, R) \in \mathcal{L},\ f^* - f_z \leq w(F(\mathbf{Z})) \leq a \times w(\mathbf{Z})^\alpha, \tag{4.6}$$

avec $F(\mathbf{Z})$ une fonction d'inclusion de f sur $\mathbf{Z}$ et a et α deux réels strictement positifs.

Si $\overline{F(\mathbf{Z})} \leq f^*$, le sous-domaine $\mathbf{Z}$ peut alors être supprimé de $\mathcal{L}$, car il ne doit pas vérifier les contraintes, puisque f^* est la valeur du minimum global ; et si $\underline{F(\mathbf{Z})} > f^*$, la phase d'élimination de la ligne 14 de l'algorithme 10 devra éliminer le sous-domaine $\mathbf{Z}$ de $\mathcal{L}$. En effet, $\forall(\mathbf{Z}, f_z, R) \in \mathcal{L},\ f^* - f_z \leq w(F(\mathbf{Z}))$. C'est pourquoi, dans le pire des cas, si nous voulons majorer $\left(f^* - \min_{(\mathbf{Z}, f_z, R)\in\mathcal{L}} f_z\right)$ par ϵ_f, nous devons majorer $a \times w(\mathbf{Z})^\alpha$ par ϵ_f. On obtient donc que la formule (4.5) du critère d'arrêt s'écrit :

$$w(\mathbf{Z}) \leq \left(\frac{\epsilon_f}{a}\right)^{\frac{1}{\alpha}}. \tag{4.7}$$

Théorème 4.2.11 *En utilisant le critère d'arrêt sur la précision de la valeur du minimum global, un majorant du nombre maximal d'itérations de la boucle principale de* IBBA-LM$^{\texttt{bf}}$*, noté dans ce cas* IBBA-LM$^{\texttt{bf}}_{\epsilon_f}$*, est :*

$$\mathcal{N}(\text{IBBA-LM}^{\texttt{bf}}_{\epsilon_f}(\boldsymbol{X})) = \texttt{MaxElts} \times \left(\sum_{i=1}^{n} \left\lceil \log_2 \frac{w(\mathbf{x}_i)}{\left(\frac{\epsilon_f}{a}\right)^{\frac{1}{\alpha}}} \right\rceil\right),$$

et l'ordre de complexité en temps correspondant est :

$$\mathcal{O}(\text{IBBA-LM}^{\texttt{bf}}_{\epsilon_f}(\boldsymbol{X})) = n \times P_{IBBA\text{-}LM^{\texttt{bf}}_{\epsilon_\mathcal{L}}}(n),$$

pour être plus précis, cet ordre de grandeur doit être multiplié par une constante qui peut prendre une grande valeur : $\texttt{MaxElts} \times \left\lceil \log_2 \frac{w(\boldsymbol{X})}{\left(\frac{\epsilon_f}{a}\right)^{\frac{1}{\alpha}}} \right\rceil$.

Démonstration. Cette proposition est directement obtenue à partir de l'équation (4.7) et en appliquant les théorèmes 4.2.7 et 4.2.8. ∎

4.3 Expériences numériques

Dans cette section, nous considérons notre algorithme nommé *IBBA-LM*$_{Hp}$. Celui-ci combine *IBBA* (algorithme 10) et l'heuristique *Hp* décrite dans l'algorithme 12 et la section 4.1. Sa structure de données $\mathcal{L}$ est triée par ordre croissant des minorants f_z de chaque élément et le critère d'arrêt utilisé est celui sur l'encadrement de la valeur du minimum global, voir équation (4.5).

Cette section est divisée en deux. La première partie étudie un exemple de Hang Tuy correspondant à l'exemple *hs071* de la librairie 2 du site COCONUT [81]. Nous discuterons du comportement de cette nouvelle approche *IBBA-LM*$_{Hp}$ en faisant notamment varier la limite `MaxElts` sur la mémoire. Dans la seconde partie, nous testerons l'efficacité de *IBBA-LM*$_{Hp}$ sur un ensemble de problèmes tests d'optimisation globale avec contraintes issues de la librairie 1 de COCONUT [81, 83, 95], ayant entre 11 et 33 variables.

Tous ces tests sont réalisés sur un ordinateur PC-Intel-Xeon-3GHz muni de 2 Go de mémoire vive et utilisant un système Linux 64-bit. Le code implémentant les algorithmes est complètement écrit en Fortran 90/95 et compilé avec le compilateur *f90* de SUN incluant une librairie pour l'arithmétique d'intervalles.

4.3.1 L'exemple *hs071*

Voici l'exemple *hs071* de la librairie 2 du site COCONUT [81] :

$$\begin{cases} \min\limits_{x\in[1;5]^4} & x_3 + (x_1 + x_2 + x_3)x_1x_4 \\ s.t. & x_1^2 + x_2^2 + x_3^2 + x_4^2 = 40, \\ & x_1x_2x_3x_4 \geq 25. \end{cases} \tag{hs071}$$

Dans le tableau 4.1, nous résolvons le problème (hs071) en utilisant *IBBA* (algorithme 10) avec une précision de $\epsilon_f = 10^{-8}$ demandée sur la valeur du minimum global et sur les contraintes. Nous y comparons les résultats obtenus par *IBBA-LM*$_{Hp}$ en utilisant une limite `MaxElts` sur le nombre d'éléments dans $\mathcal{L}$ prenant les valeurs 250 000, 500 000 et 750 000, ce qui correspond approximativement à l'espace de mémoire vive disponible indiqué en méga-octet à la ligne 'RAM' du tableau 4.1. *Nits* corresponds au nombre d'itérations de la boucle principale de l'algorithme 10. La variable P représente la précision certifiée sur l'encadrement du minimum global à la fin de l'algorithme *IBBA-LM*$_{Hp}$, c'est-à-dire la valeur du minimum global est certifiée appartenir à l'encadrement $[\tilde{f} - P, \tilde{f}]$.

Algo. `MaxElts`	*IBBA-LM*$_{Hp}$ 250 000	*IBBA-LM*$_{Hp}$ 500 000	*IBBA-LM*$_{Hp}$ 750 000	*IBBA* (pas de limite)
max$\lvert\mathcal{L}\rvert$	250 000	500 000	750 000	$> 4\,000\,000$
RAM	128Mo	256Mo	512Mo	2Go
Nits	1 698 543	3 232 935	4 968 019	22 710 768
Temps	$1.72min$	$3.63min$	$7.28min$	$46.96min$
Solution	17.0140175910	17.0140173514	17.014017320914	17.0140172872
P	6.0089×10^{-6}	1.652×10^{-6}	7.05676×10^{-7}	1×10^{-8}

TABLE 4.1 – Comportement de l'algorithme *IBBA-LM*$_{Hp}$ sur l'exemple *hs071*.

Dans le tableau 4.1 nous remarquons que :

– Si la limite sur la mémoire est très basse `MaxElts` = 250 000, *IBBA-LM*$_{Hp}$ résout déjà le problème avec une précision satisfaisante inférieure à 10^{-5} et un temps inférieur à 2 minutes. Ainsi, cela montre que sur cet exemple, un encadrement fin sur la valeur du minimum global peut être obtenu sans avoir besoin de beaucoup de mémoire vive.
– Sur cet exemple, *IBBA* a besoin de 2 Go de mémoire, ce qui est exactement la limite physique de notre ordinateur. C'est pourquoi le système d'exploitation peut swapper une partie de $\mathcal{L}$ sur le disque dur et les performances de calcul s'en voient dégradées (il existe un facteur 1 000 entre un accès à la mémoire vive et un accès sur le disque dur). Pour cet exemple, il est possible qu'il y ait des swaps sur le disque dur aux vues des 47 minutes nécessaires à la résolution du problème complet.

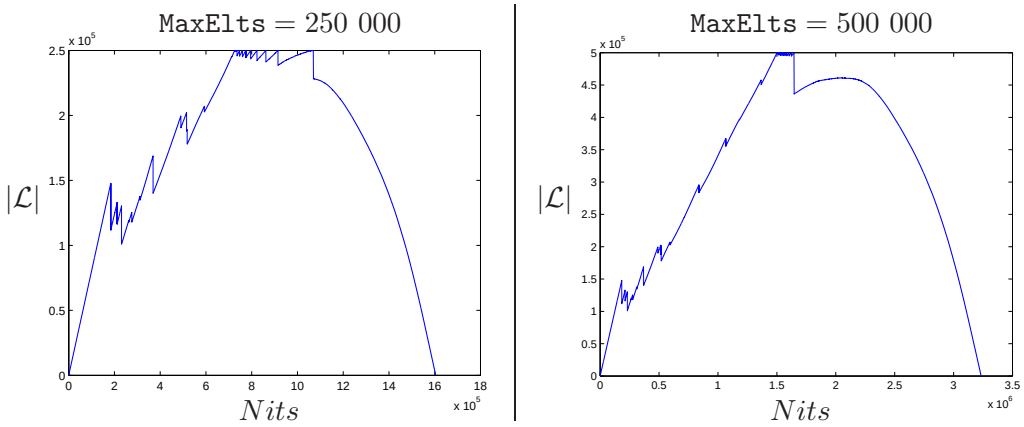


FIGURE 4.2 – Comportement de *IBBA-LM*$_{Hp}$ sur l'exemple *hs071*.

Dans la figure 4.2, nous représentons le comportement de *IBBA-LM*$_{Hp}$ en limitant la taille de $\mathcal{L}$ à 250 000 et 500 000 éléments sur l'exemple (hs071) ; le nombre d'éléments stockés dans $\mathcal{L}$ est dessiné en fonction du nombre d'itérations $Nits$. On peut remarquer :

– Dans la première partie des courbes, on peut noter que la taille de $\mathcal{L}$ croît en suivant approximativement une loi exponentielle.
– Quelques aspérités apparaissent sur les courbes lorsque qu'une meilleure solution réalisable est trouvée, ce qui permet d'enlever des éléments stockés inutilement dans $\mathcal{L}$.
– Lorsque la limite `MaxElts` est atteinte, le nombre d'éléments stockés reste proche de la limite. La variable P de l'algorithme 12 est alors incrémentée plusieurs fois de suite.
– Enfin, la convergence de l'algorithme est renforcée une fois que $\tilde{f} - P$ a atteint une valeur acceptable, c'est-à-dire une valeur qui permet à l'algorithme de converger sans atteindre une fois de plus la limite `MaxElts`.

4.3.2 Tests numériques de la librairie 1 de COCONUT

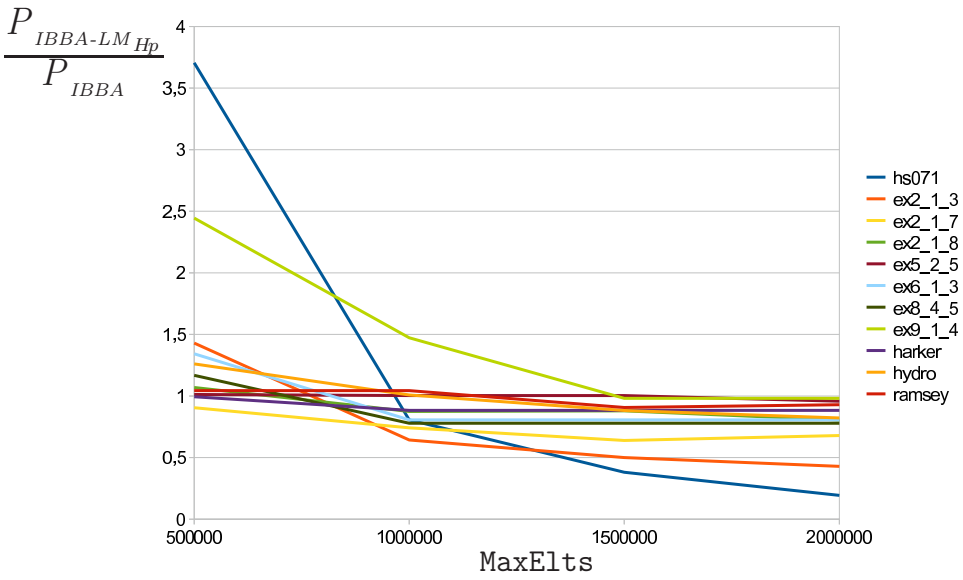
Afin d'illustrer le comportement de notre algorithme *IBBA-LM*$_{Hp}$, nous avons choisi quelques exemples issus de la librairie 1 de COCONUT [81]. Leurs nombres de variables fluctuent entre 11 et 33, ce qui est déjà assez important pour un algorithme de Branch and Bound par intervalles. Pour concentrer notre étude sur l'heuristique *Hp*, nous donnons à *IBBA-LM*$_{Hp}$ la meilleure solution connue dès le début de l'algorithme 10 (en initialisant la variable $\tilde{f}$ avec cette valeur à la place de $+\infty$) i.e. au début de *IBBA-LM*$_{Hp}$, $\tilde{f}$ correspond à la meilleure solution réalisable connue, ayant été précédemment trouvée par des algorithmes tels que BARON, MINOS, DONLP2 ou LINGOS, voir [81]. La valeur de ces solutions non encore certifiées est reportée dans la colonne *BestSol* du tableau 4.2. Cette approche peut être facilement généralisable : une bonne solution peut être obtenue en utilisant un autre algorithme (par exemple une recherche Tabou ou un VNS) et *IBBA* sera ensuite utilisé pour valider la solution ou l'améliorer. Ainsi, l'objectif n'est pas de résoudre complètement le problème ou de trouver la meilleure solution, mais plutôt de savoir quelle est la plus petite précision pouvant être garantie de façon fiable et robuste avec la mémoire disponible d'un ordinateur quelconque.

Dans le tableau 4.2, tous les résultats des tests sont présentés. N désigne le nombre de variables du problème test et M son nombre de contraintes. P représente la précision certifiée obtenue à la fin de l'algorithme *IBBA-LM*$_{Hp}$, ce qui signifie que lorsque l'algorithme termine, la valeur du minimum global appartient avec certitude à $[BestSol - P, BestSol]$ (pour tous les exemples la solution fournie $BestSol$ n'a pas été améliorée). Nous avons réalisé plusieurs tests sur chaque exemple en changeant la valeur de `MaxElts`, ainsi qu'une exécution sans l'heuristique *Hp*. Nous avons choisi $\tilde{f} - \min_{(\mathbf{z},f_z,R)\in\mathcal{L}} f_z \leq \epsilon_f$ comme critère d'arrêt. Et nous démarrons l'algorithme avec $\epsilon_f = 10^{-6}$ excepté pour l'exemple (hs071) avec $\epsilon_f = 10^{-8}$. Nous avons ajouté un critère d'arrêt sur le temps de calcul maximal fixé à 60 minutes. Pour l'algorithme *IBBA* sans *Hp*, celui-ci s'arrête également si la taille de $\mathcal{L}$ dépasse les 2 000 000 éléments. Si l'algorithme s'arrête avec l'un des deux tests d'arrêt précédents, la valeur de P affichée dans le tableau 4.2 est obtenue par : $\max\{P,\ \tilde{f} - \min_{(\mathbf{z},f_z)\in\mathcal{L}} f_z\}$.

Dans le tableau 4.2, on peut noter que :

- La précision P est meilleure lorsque la valeur de `MaxElts` croît, excepté sur quelques tests dû au critère d'arrêt sur le temps de calcul.
- Pour tous les exemples, *IBBA* et *IBBA-LM*$_{Hp}$ ont exactement le même comportement jusqu'à ce que la taille de $\mathcal{L}$ atteigne `MaxElts`, voir proposition 4.2.6. Lorsque $|\mathcal{L}| =$ `MaxElts` $=$ 2 000 000, *IBBA* sans *Hp* doit s'arrêter, car il n'y a plus de mémoire disponible sur l'ordinateur, alors que *IBBA-LM*$_{Hp}$ peut continuer à améliorer la valeur de la précision certifiée P.
- Pour obtenir une précision certifiée P équivalente à celle obtenue avec *IBBA* sans *Hp*, *IBBA-LM*$_{Hp}$ n'a besoin de stocker que 1 500 000 éléments. Ainsi, avec moins de mémoire on peut obtenir autant d'information.
- Pour tous les exemples, *IBBA* sans *Hp* atteint très rapidement la limite des 2 000 000 d'éléments, généralement cela nécessite moins de 15 minutes.

FIGURE 4.3 – Visualisation de l'amélioration de la précision certifiée.



La figure 4.3 illustre clairement l'intérêt réel de l'heuristique *Hp*. Sur le graphique, on peut y voir, pour chaque problème test du tableau 4.2, la valeur de `MaxElts` en abscisse et en ordonné le rapport $\frac{P_{IBBA\text{-}LM_{Hp}}}{P_{IBBA}}$, où P_{IBBA} est la précision certifiée obtenue avec *IBBA* sans heuristique et $P_{IBBA\text{-}LM_{Hp}}$ prend les valeurs des précisions certifiées obtenues avec $IBBA\text{-}LM_{Hp}$ pour chaque valeur de `MaxElts`.

Comme on peut le remarquer, pratiquement toutes les courbes descendent très rapidement en dessous de la valeur 1, ce qui signifie que la précision obtenue avec $IBBA\text{-}LM_{Hp}$ est meilleure qu'avec *IBBA*. Ainsi, on obtient de meilleurs résultats avec l'heuristique et `MaxElts` = 1 500 000 que sans heuristique et une mémoire pouvant stocker 2 000 000 éléments.

4.4 Discussion

Dans cette étude, le code *IBBA* que nous avons utilisé est très simple. Il ne s'agit en effet que de l'algorithme 10 avec comme unique technique d'accélération la propagation des contraintes décrite dans la section 1.3.2. Les encadrements sont simplement calculés par l'extension naturelle aux intervalles. Ainsi, pour produire un algorithme plus efficace, il suffirait d'inclure les techniques d'accélération étudiées au chapitre 3, d'autres techniques de calcul d'encadrement comme celles décrites au chapitre 2 ou encore d'autres heuristiques que celles proposées dans la section 4.1.

Ainsi, toutes les techniques développées dans les chapitres précédents sont cumulables avec l'approche métaheuristique que nous venons d'étudier.

TABLE 4.2 – Comportement de *IBBA-LM $_{Hp}$* sur plusieurs problèmes issus du site COCONUT.

			MaxElts	*IBBA-LM $_{Hp}$* 500 000		*IBBA-LM $_{Hp}$* 1 000 000		*IBBA-LM $_{Hp}$* 1 500 000		*IBBA-LM $_{Hp}$* 2 000 000		*IBBA* sans *Hp* 2 000 000	
Nom	N	M	*BestSol*	*P*	T(min)	*P*	T(min)	*P*	T(min)	*P*	T(min)	*P*	T(min)
hs071	4	2	17.01401728	1.6520e-6	3.63	3.59487e-7	7.57	1.70085575e-7	10.71	8.62882376e-8	32.26	4.45850297e-7	6.03
ex2_1_3	13	9	-15.0	2.0	2.48	0.9	11	0.7	30.33	0.6	47.08	1.3984375	1.75
ex2_1_7	20	10	-4150.4101	610.0	22.47	500.0	42.8	430.957	60	457.7148	60	673.814200831	16.68
ex2_1_8	24	10	15639.0	11000.0	6.97	9000.0	39.55	9047.25	60	8280.4375	60	10283.4375	3.61
ex5_2_5	32	19	-3500	12100.0	15.03	12000.0	21.18	12000.0	24.03	11450	60	11950	8.81
ex6_1_3	13	9	-0.3525	0.5	2.48	0.3	7.95	0.3	25.28	0.3	13.13	0.37225180	2.93
ex8_4_5	15	11	3.e-4	3.e-5	3.76	2.e-5	12.05	2.e-5	25.28	2.e-5	32.73	2.56827015e-5	3.5
ex9_1_4	11	9	-37	4.97951e-5	1.25	3.e-5	2.93	2.e-5	8.33	2.e-5	9.88	2.03741893e-5	1.75
harker	20	7	-986.51350	900.0	7.22	800.0	33.46	800.0	44.55	800.0	31.91	905.61489392	3.71
hydro	31	24	4366944.0	200000.0	5.72	160000.0	19.38	140000.0	43.55	130303.5687	60	158596.7221	5.25
ramsey	33	22	-2.48750	0.07	11.22	0.07	11.33	0.06083178	60	0.06234392	60	0.06707264	7.06

Chapitre 5

Applications

Depuis le début du développement de notre algorithme *IBBA* avec la thèse de Frédéric Messine [68], celui-ci n'a cessé d'être amélioré. Mais parallèlement, les problèmes se sont de plus en plus complexifiés, atteignant toujours la limite des capacités de notre algorithme.

Dans cette partie, nous étudierons la résolution de problèmes issus du génie électrique ainsi que quelques problèmes ouverts de géométrie plane.

5.1 Étude des polygones convexes équilatéraux de largeur unitaire

5.1.1 Introduction

Deux familles de problèmes extrémaux sur les polygones convexes sont présentées dans [9, 11]. Pour un nombre fixe de côtés, chaque famille consiste à maximiser ou minimiser un attribut tel que l'aire, le périmètre, le diamètre, la somme des distances entre toutes les paires de sommets ou la largeur d'un polygone en fixant un autre attribut. Rappelons que le diamètre d'un polygone est la longueur du plus grand segment joignant deux sommets, et sa largeur est la distance minimale entre deux droites parallèles qui encadrent le polygone. Dans la deuxième famille de problèmes, les polygones sont nécessairement équilatéraux, alors que dans la première famille, la longueur des côtés est libre. Ces deux articles donnent des solutions ou fournissent les références de solutions de la plupart des problèmes, en indiquant ceux qui sont triviaux, et ceux qui restent encore ouverts.

Dans cette section, nous nous intéresserons à plusieurs questions ouvertes sur la maximisation du périmètre, du diamètre, de la somme des distances et de l'aire d'un polygone convexe et équilatéral de largeur unitaire. Les sections 5.1.1, 5.1.2, 5.1.3 et 5.1.4 reprennent des éléments de l'article [14].

Énoncé des solutions optimales

La question de la minimisation du périmètre et du diamètre d'un polygone convexe de largeur unitaire a déjà été étudiée dans [12] et [19], respectivement. Pour ces deux problèmes, tout comme pour le problème de maximisation du périmètre d'un polygone convexe à diamètre unitaire [10, 13, 28, 39, 105], les polygones réguliers à base Reuleaux, nommés *clipped-Reuleaux regular polygons* [92], sont optimaux lorsque le nombre de côtés contient un facteur impair. Les polygones réguliers à base Reuleaux sont équilatéraux, et c'est pourquoi ils sont aussi optimaux pour le cas équilatéral. Néanmoins, les cas où le nombre de côtés est une puissance de 2 restent ouverts.

Mossinghoff [80] appliqua les résultats de Pál [87] pour montrer que lorsque le nombre de côtés est un multiple de 3, alors le polygone convexe de largeur unitaire qui minimise l'aire est arbitrairement proche du triangle équilatéral. Lorsque le nombre de côtés n'est pas un multiple de 3, le problème reste encore ouvert.

Dans le cas où les polygones convexes de largeur unitaire ne sont pas restreints aux équilatéraux, les questions de maximiser le périmètre P_n, l'aire A_n, la somme des distances S_n ou le diamètre D_n sont triviales, puisqu'ils peuvent être arbitrairement grands lorsque la longueur des côtés c_n tend vers $+\infty$. Dans le cas équilatéral lorsque le nombre n de côtés est pair, ces attributs peuvent encore être arbitrairement grands comme l'illustre la partie gauche de la figure 5.1 avec un parallélogramme.

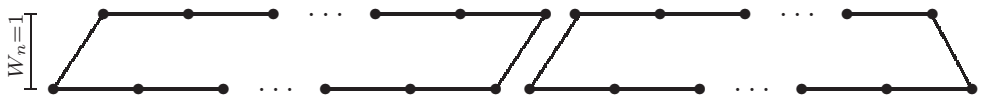


FIGURE 5.1 – Polygones convexes équilatéraux à n côtés et de largeur unitaire

Cette section étudie les problèmes de maximisation non triviaux, lorsque le nombre de côtés équilatéraux $n = 2m+1$ est impair. La proposition suivante montre que la valeur de l'optimum est bornée.

Proposition 5.1.1 *Lorsque n est impair, les trois polygones convexes équilatéraux à n côtés qui maximisent le périmètre, le diamètre et l'aire ont des dimensions finies.*

Démonstration. Considérons le problème équivalent de minimiser la largeur d'un polygone convexe équilatéral à périmètre unitaire, diamètre unitaire ou aire unitaire. Supposons que le polygone optimal de ces problèmes a une largeur nulle. Les sommets du polygone sont donc colinéaires. Or, ceci est impossible puisque le nombre de côtés est impair et que tous les côtés sont égaux.

Ainsi, le périmètre, le diamètre et l'aire de tout polygone convexe équilatéral à largeur unitaire et à $2m + 1$ coté, sont bornés. ∎

Le cas où $n = 3$ est trivial puisque le triangle équilatéral de largeur unitaire est unique. Pour $n = 2m + 1 \geq 5$, nous montrons que pour les quatre problèmes considérés, les solutions optimales sont arbitrairement proches d'un trapèze dont les côtés non parallèles ont la même longueur égale à $\hat{c}_n = \frac{2}{\sqrt{3}}$, et les côtés parallèles comme longueur $\frac{2}{\sqrt{3}}m$ et $\frac{2}{\sqrt{3}}(m-1)$. Trois des problèmes sont prouvés mathématiquement et un conjecturé avec une preuve numérique à 10^{-10} pour $n = 5,\ 7$ et 9. Le mot *arbitrairement proche* est utilisé pour indiquer que techniquement le trapèze a 4 côtés et non $2m + 1$. Ce trapèze, noté $\hat{T}_n$, est illustré dans la partie droite de la figure 5.1, avec son périmètre $\hat{P}_n$, son diamètre $\hat{D}_n$, sa somme des distances $\hat{S}_n$ et son aire $\hat{A}_n$.

Résultats généraux pour les polygones convexes équilatéraux de largeur unitaire

Considérons un polygone convexe équilatéral à n côtés ayant une largeur W_n strictement positive. Les coordonnées sont représentées sur un plan orthonormé xy, et toutes les distances sont euclidiennes. Soit $y = 0$ et $y = W_n$ deux droites parallèles encadrant le polygone, et définissant sa largeur. En appliquant une symétrie verticale si nécessaire, supposons que la droite $y = 0$ contient au moins un côté du polygone et la droite $y = W_n$ au moins un sommet.

Soit A un sommet du polygone sur la droite $y = 0$ ayant la plus petite abscisse, et B le sommet sur la droite $y = W_n$ ayant la plus petite abscisse. Soit C un sommet du polygone ayant la plus grande abscisse (s'il y en a plusieurs, on considérera celui ayant la plus grande ordonnée). Les coordonnées de A, B et C sont notées (x_A, y_A), (x_B, y_B) et (x_C, y_C), respectivement. Soit B^- le sommet adjacent à B dans le sens inverse des aiguilles d'une montre. Ces 4 sommets sont représentés sur la figure 5.2, et seront utilisés dans toutes les figures suivantes.

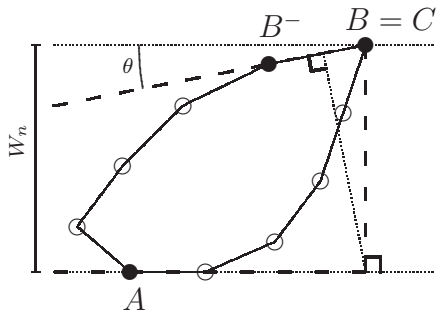


FIGURE 5.2 – Cas où le sommet de la droite $y = W_n$ de plus petite abscisse coïncide avec le sommet ayant la plus grande abscisse.

Premièrement, démontrons que $A \neq B \neq C \neq A$. Par construction, $A \neq C$ puisqu'il existe au moins un sommet sur la droite $y = 0$ qui est adjacent à A avec une abscisse plus grande. De plus, $A \neq B$ car $y_B = W_n > 0 = y_A$.

Maintenant, supposons par contradiction que $B = C$. Ceci implique qu'il existe un unique sommet sur la droite $y = W_n$ comme l'illustre la figure 5.2. Soit $\theta \in]0, \frac{\pi}{2}]$ l'angle strictement positif entre la droite $y = W_n$ et la droite (BB^-). Par convexité du polygone, tous ces sommets sont contenus dans le triangle formé par les droites $y = 0$, $x = x_C$ et (BB^-). Il est représenté par des pointillés sur la figure 5.2. Or, la largeur de ce triangle est égale à $W_n \cos\theta$ ce qui est strictement inférieur à W_n. Ceci contredit le fait que W_n soit la largeur minimale du polygone. Ainsi, nous avons montré que $B \neq C$.

Ces trois points distincts A, B et C sont utilisés dans les preuves des sections suivantes.

5.1.2 Maximiser le périmètre et le diamètre

Cette section démontre les résultats annoncés dans la section 5.1.1. Les polygones convexes équilatéraux de largeur unitaire qui maximisent le périmètre et le diamètre sont identiques et arbitrairement proches du trapèze $\hat{T}_n$. Le théorème 5.1.3 démontre le résultat de la maximisation du périmètre et le corollaire 5.1.4, celui du diamètre.

Tous les résultats sont présentés en se basant sur les notations de la section 5.1.1.

Lemme 5.1.2 *Dans tout polygone convexe à* $(2m+1)$ *côtés unitaires et de largeur minimale, le sommet avec* $y > 0$ *adjacent à* A *est* B.

Démonstration. Considérons un polygone convexe à $(2m+1)$ côtés unitaires et de largeur minimale W_n^*, et soit A^+ le sommet avec $y > 0$ adjacent à A.

Supposons par contradiction que $A^+ \neq B$. Notons les angles intérieurs du polygone α, β, γ et α^+ associés respectivement aux sommets A, B, C et A^+. La définition de ces sommets nous assure pour ces angles les inégalités strictes suivantes : $\alpha < \pi$, $\beta < \pi$, $\gamma > 0$ et $\alpha^+ > 0$. Le quadrilatère AA^+BC et les quatre angles sont illustrés dans la figure 5.3.

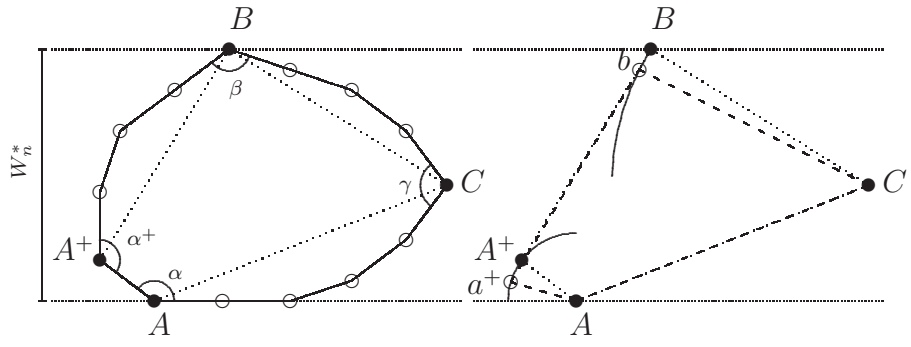


FIGURE 5.3 – Le quadrilatère AA^+BC inscrit dans le polygone.

La démonstration consiste à construire un nouveau polygone convexe à $(2m+1)$ côtés unitaires de même périmètre, mais avec une largeur inférieure à l'original. La

construction est basée sur une transformation du quadrilatère AA^+BC en Aa^+bC. Afin de faire cela, notons b le point du plan xy satisfaisant $\|bC\| = \|BC\|$ et dont l'ordonnée de b est légèrement inférieure à celle de B, et notons a^+ un autre point proche de A^+ satisfaisant $\|Aa^+\| = \|AA^+\|$ et $\|a^+b\| = \|A^+B\|$. Les points a^+ et b sont illustrés dans la partie droite de la figure 5.3, et sont localisés sur les arcs circulaires centrés en A et C.

Pour créer le nouveau polygone basé sur Aa^+bC, on procède ainsi dans le sens des aiguilles d'une montre à partir de C :

- On recopie tous les sommets entre C et A inclu.
- On considère la transformation qui préserve les distances et qui transforme A^+ en a^+ et B en b, puis en leur appliquant cette transformation, on ajoute tous les sommets entre A^+ et B inclus.
- On considère la rotation de centre C qui transforme B en b, et on l'applique à tous les sommets entre B et C.

Observons que les ordonnées de tous les sommets du nouveau polygone sont strictement inférieures à W_n^*, à l'exception peut-être de l'ordonnée y_C de C.

Par construction, le nouveau polygone est équilatéral, avec le même périmètre que l'original, puisque toutes les transformations utilisées conservent les distances. La construction décroît la valeur des angles $\gamma > 0$ et $\alpha^+ > 0$ et croît celle de $\alpha < \pi$ et $\beta < \pi$. Tous les autres angles restent inchangés. En prenant le sommet b suffisamment proche de B, ces inégalités restent valides pour le nouveau polygone, ce qui prouve qu'il est convexe.

Rappelons que la largeur du nouveau polygone ne peut être inférieure à la valeur minimale de W_n^*, ce qui implique nécessairement que $y_C = W_n^*$. Or, le résultat issu de la section 5.1.1 et illustré dans la figure 5.2 montre que la largeur du nouveau polygone est strictement inférieure à W_n^*, ce qui aboutit à une contradiction.
Ainsi, $A^+ = B$. ∎

Le lemme précédent assure que dans un polygone optimal, il y a un sommet adjacent à A tel que $y = W_n^*$. Ce résultat réduit le nombre des configurations possibles et aboutit à la preuve de notre résultat principal.

Théorème 5.1.3 *Le périmètre de tout polygone convexe équilatéral à $(2m+1)$ côtés et de largeur unitaire est majoré par $\frac{2}{\sqrt{3}}(2m+1)$. Cette borne est atteinte à la limite par des polygones arbitrairement proches du trapèze $\hat{T}_n$.*

Démonstration. Au lieu de maximiser le périmètre sur un polygone de largeur unitaire, la démonstration aborde la question équivalente de montrer que la largeur de tout polygone convexe équilatéral à $(2m+1)$ côtés avec un périmètre de $\hat{P}_n = \frac{2n}{\sqrt{3}}$ est supérieure à 1, et que cette borne inférieure est atteinte à la limite par des polygones arbitrairement proches du trapèze $\hat{T}_n$.

Considérons un polygone convexe équilatéral à $(2m+1)$ côtés avec un périmètre de $\hat{P}_n$ et une largeur minimale $W_n^* > 0$. Le lemme 5.1.2 assure que le sommet avec $y > 0$ adjacent à A est B. Notons Z le sommet du polygone avec la plus grande abscisse sur la droite $y = 0$. Par symétrie avec le sommet A, le lemme 5.1.2 assure

que le sommet avec $y > 0$ adjacent à Z est sur la droite $y = W_n^*$. Ainsi, par convexité du polygone, le polygone convexe équilatéral à $(2m+1)$ côtés avec un périmètre de $\hat{P}_n$ et une largeur minimale W_n^* est un trapèze, un parallélogramme ou un triangle.

Notons p le nombre de côtés du polygone sur la droite $y = W_n^*$ et q le nombre de côtés sur la droite $y = 0$. En appliquant une symétrie verticale si nécessaire, supposons que $p \leq q$, les deux chemins de sommets consécutifs joignant A et Z satisfont $q \leq p+2$. C'est pourquoi, le nombre de côtés q ne peut prendre qu'une des valeurs suivantes : $p, p+1$ ou $p+2$. Mais puisqu'il y a au total $p+q+2$ côtés, et que ce nombre est impair, il s'en déduit que la seule possibilité est $q = p+1$. C'est pourquoi, le polygone convexe équilatéral à $(2m+1)$ côtés avec un périmètre de $\hat{P}_n$ et une largeur minimale W_n^* est le trapèze $\hat{T}_n$ et donc $W_n^* = \hat{W}_n = 1$. ∎

Le dernier théorème implique que le périmètre maximal d'un polygone convexe équilatéral à n côtés et de largeur unitaire est $\hat{P}_n = \frac{2}{\sqrt{3}}(2m+1)$, lorsque $n = 2m+1$. Le résultat similaire sur le diamètre s'ensuit directement comme corollaire.

Corollaire 5.1.4 *Le diamètre de tout polygone convexe équilatéral à $(2m+1)$ côtés et de largeur unitaire est majoré par $\frac{2m}{\sqrt{3}}$. Cette borne est atteinte à la limite par des polygones arbitrairement proches du trapèze $\hat{T}_n$.*

Démonstration. Pour $n = 2m+1$, notons P_n le périmètre d'un polygone convexe équilatéral de largeur unitaire. Les extrémités de toutes diagonales joignant deux sommets sont nécessairement jointes par un chemin d'au plus $\lfloor \frac{n}{2} \rfloor = m$ sommets consécutifs. En combinant ceci avec le fait que la longueur de chaque côté est $\frac{P_n}{n}$ et que le théorème 5.1.3 fournit une borne supérieure, on obtient une borne supérieure du diamètre maximal du polygone : $\frac{\hat{P}_n}{n} \times m = \frac{2}{\sqrt{3}} m$. Cette borne est atteinte par le trapèze $\hat{T}_n$ présentée dans la figure 5.1. ∎

5.1.3 Maximisation de l'aire

Tout au long de cette section, nous noterons A_n^* l'aire maximale d'un polygone convexe équilatéral de largeur unitaire. Notons P_n et D_n le périmètre et le diamètre d'une solution optimale, et fixons $c_n = \frac{P_n}{n}$ la longueur de ces côtés. Les résultats précédents nous prouvent que :

$$P_n \leq \hat{P}_n, \quad D_n \leq \hat{D}_n \quad \text{et} \quad \hat{A}_n \leq A_n^*.$$

Lemme 5.1.5 *Soit $n = 2m+1 \geq 5$ un nombre impair et considérons un polygone convexe équilatéral de largeur unitaire avec une aire maximale A_n^*, et soit c_n la longueur de chacun de ces côtés. Alors, la longueur vérifie $c_n > 1$, et si $n \geq 7$, $c_n \geq \frac{1}{14}\left(\frac{20}{\sqrt{3}} + \pi\right)$.*

Démonstration. Jaglom et Boltianskii [108, Problème 6.11] montrent que l'aire maximale d'une figure convexe de largeur unitaire et de périmètre P_n est $\frac{2P_n - \pi}{4}$.

En particulier, ceci fournit un majorant de A_n^*. En combinant ceci avec la borne inférieure de $\hat{A}_n$, on obtient :

$$\frac{2m-1}{\sqrt{3}} \;=\; \hat{A}_n \leq A_n^* \;\leq\; \frac{2P_n - \pi}{4} \;=\; \frac{2(2m+1)c_n - \pi}{4}.$$

En résolvant cette équation pour c_n, on obtient l'inégalité suivante :

$$c_n \;\geq\; \frac{8m + \sqrt{3}\pi - 4}{2\sqrt{3}(2m+1)}.$$

Cette borne inférieure sur c_n est monotone croissante en fonction de m. La valeur la plus basse se produit pour $n = 2m+1 = 5$, et donne un minorant légèrement supérieur à 1. Lorsque $n = 2m+1 \geq 7$, le minorant est $\frac{1}{14}\left(\frac{20}{\sqrt{3}} + \pi\right)$. ∎

La preuve que le trapèze $\hat{T}_n$ réalise l'aire maximale est faite en deux parties. Le cas du pentagone est d'abord prouvé avec le lemme suivant, et les cas restants ($n \geq 7$) sont prouvés par le théorème 5.1.7.

Lemme 5.1.6 *L'aire de tout pentagone convexe équilatéral de largeur unitaire est majorée par $\sqrt{3}$. Cette borne est atteinte à la limite par des pentagones arbitrairement proches du trapèze $\hat{T}_5$.*

Démonstration. Soit A, B et C les sommets définis dans la section 5.1.1 sur un pentagone convexe optimal de largeur unitaire avec des côtés de longueur c_5. Les quatre configurations possibles des positions relatives de A et B sont représentées sur la figure 5.4 : sur les deux figures de gauche, l'abscisse de B est entre x_A et $x_A + c_5$, et sur les figures de droite, elle est inférieure à x_A. Sur les deux figures du haut, le sommet B est adjacent à A, et pas dans celles du bas.

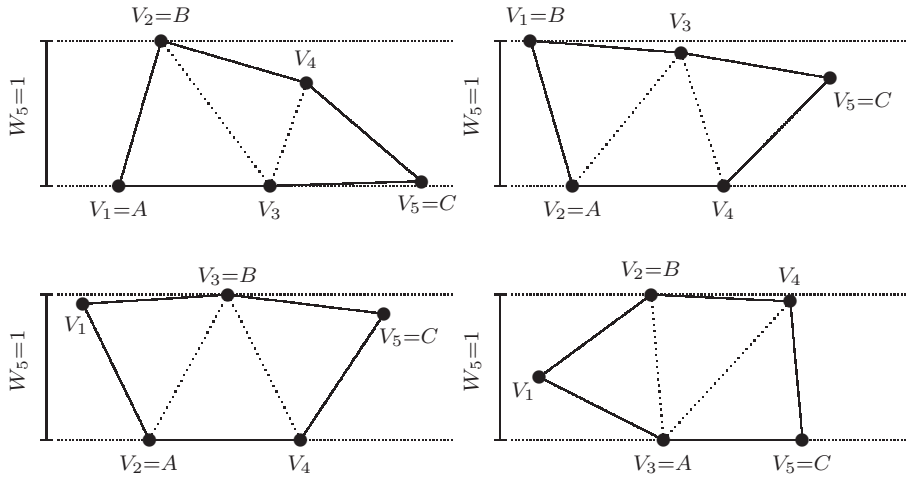


FIGURE 5.4 – Les quatre configurations possibles pour le pentagone.

Dans les quatre configurations représentées dans la figure 5.4, les sommets sont notés par $V_i = (x_i, y_i)$ avec $i = 1, 2, \ldots, 5$ et sont ordonnés de tel sorte que $x_1 < x_2 < \cdots < x_5$. Cet ordre avec des inégalités strictes est possible puisque le lemme 5.1.5 assure que la longueur c_5 des côtés du pentagone est strictement plus grand que 1. Observons que pour $i \in \{2, 3, 4\}$, chaque triangle $V_{i-1}V_iV_{i+1}$ dans chaque configuration a une aire égale à $\frac{h_i c_5}{2}$ avec h_i la hauteur associée au sommet V_i. Néanmoins, puisque $h_i \leq \max(|y_i - y_{i-1}|, |y_i - y_{i+1}|) \leq 1$ pour tout $i \in \{2, 3, 4\}$, il s'en déduit que l'aire du pentagone satisfait :

$$A_5^* \;=\; \frac{c_5}{2}(h_2 + h_3 + h_4) \;\leq\; \frac{3c_5}{2} \;\leq\; \frac{3\hat{c}_5}{2} \;=\; \sqrt{3}. \tag{5.1}$$

Ce majorant est atteint par le trapèze $\hat{T}_5$. ∎

Le théorème suivant clôture le cas général avec un nombre impair de sommets.

Théorème 5.1.7 *L'aire de tout polygone convexe équilatéral à $(2m+1)$ côtés et de largeur unitaire est majorée par $\frac{2m-1}{\sqrt{3}}$. Cette borne est atteinte à la limite par des polygones arbitrairement proches du trapèze $\hat{T}_n$.*

Démonstration. Le cas $n = 3$ est trivial, celui de $n = 5$ a déjà été démontré dans le lemme 5.1.6. Considérons un polygone convexe de largeur unitaire, d'aire maximale A_n^* et de $n = 2m+1 \geq 7$ côtés. En utilisant les notations présentées dans la section 5.1.2, notons (x_C, y_C) les coordonnées du sommet ayant la plus grande abscisse. Notons C^- et C^+ les sommets adjacents à C.

Soit α l'aire de la région comprise entre la bande de largeur 1 encadrant le polygone, elle est composé de la partie supérieure au côté C^-C et de la partie inférieure au côté CC^+, comme représenté dans la figure 5.5 par la région hachurée.

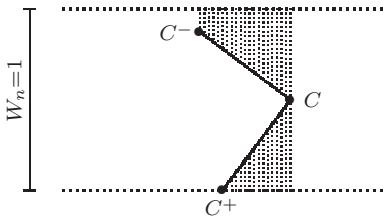


FIGURE 5.5 – Correction α du majorant de l'aire.

Étant donné que le lemme 5.1.5 prouve que $c_n > 1$ lorsque $n = 2m + 1 \geq 7$, la valeur de α est minorée par la somme de l'aire du triangle de hauteur y_C et d'hypoténuse c_n et de l'aire du triangle de hauteur $(1 - y_C)$ et d'hypoténuse c_n :

$$\alpha \geq \frac{1}{2}\left(y_C\sqrt{c_n^2 - y_C^2} + (1 - y_C)\sqrt{c_n^2 - (1 - y_C)^2}\right).$$

Ce minorant sur α est une fonction concave en fonction de y_C et elle atteint son minimum lorsque $y_C = 0$ ou 1. Dans les deux cas, l'inégalité devient $\alpha \geq \frac{1}{2}\sqrt{c_n^2 - 1}$.

Ainsi, le majorant $D_nW_n \leq mc_n$ sur la valeur de A_n^* peut être amélioré en retirant une première fois l'aire α associée au sommet C et une autre fois l'aire α associée au sommet ayant la plus petite abscisse. On aboutit aux majorants suivants :

$$\frac{2m-1}{\sqrt{3}} \leq A_n^* \leq D_nW_n - 2\alpha \leq mc_n - \sqrt{c_n^2 - 1}.$$

Rappelons que pour tout $n = 2m+1 \geq 7$, le théorème 5.1.3 et le lemme 5.1.5 fournissent des bornes sur la longueur des côtés :

$$\frac{1}{14}\left(\frac{20}{\sqrt{3}} + \pi\right) \leq c_n \leq \frac{2}{\sqrt{3}}.$$

Néanmoins, puisque $mc_n - \sqrt{c_n^2 - 1}$ est une fonction convexe en fonction de c_n, le plus grand majorant est atteint pour $c_n = \frac{1}{14}\left(\frac{20}{\sqrt{3}} + \pi\right)$ ou $c_n = \frac{2}{\sqrt{3}}$. On peut vérifier que pour $m \geq 3$, la plus grande valeur est atteinte dans le dernier cas, c'est pourquoi :

$$\frac{2m-1}{\sqrt{3}} = \hat{A}_n \leq A_n^* \leq \frac{2}{\sqrt{3}}m - \sqrt{\left(\frac{2}{\sqrt{3}}\right)^2 - 1} = \frac{2m-1}{\sqrt{3}}.$$

Ceci montre que l'aire optimale est $\frac{2m-1}{\sqrt{3}}$, et celle-ci est atteinte par le trapèze $\hat{T}_n$ illustré dans la partie droite de la figure 5.1. ∎

Notons que la dernière partie de la preuve n'est pas vraie pour $m = 2$ et c'est pourquoi le lemme 5.1.6 est nécessaire pour prouver le cas du pentagone.

5.1.4 Problèmes équivalents

On peut obtenir, par homothétie du trapèze $\hat{T}_n$, les solutions optimales des problèmes équivalents consistant à minimiser la largeur d'un polygone convexe équilatéral à périmètre, diamètre ou aire unitaire.

Le tableau 5.1 résume les solutions des cas non triviaux des problèmes équivalents prouvés dans la section précédente ainsi que les résultats de [12]. Les valeurs représentent les bornes supérieures et inférieures de la largeur d'un polygone convexe équilatéral de périmètre, diamètre ou aire unitaire.

5.1.5 Raisonnements numériques

Lorsque le raisonnement géométrique et analytique trouve ses limites, la démonstration d'un théorème peut paraître hors de portée. En cas de blocage, il est donc intéressant de considérer d'autres approches ou méthodologies de démonstration. En ce sens, l'utilisation d'un ordinateur et d'algorithmes peut pallier les limites de la preuve géométrique pure. En effet, si ces petits problèmes de géométrie ont un énoncé très court, leurs résolutions ont toutefois nécessité dans certains cas plusieurs années de recherche. C'est ce qu'illustre notamment la démonstration du problème

	$P_n = 1$	$D_n = 1$	$A_n = 1$
$\max W_n$	$\frac{1}{2n}\cot(\frac{\pi}{2n})$ Clipped-Reuleaux pour n contenant un facteur impair	$\cos(\frac{\pi}{2n})$ Clipped-Reuleaux pour n contenant un facteur impair	$\sqrt[4]{3}$ pour $n = 3m$
$\min W_n$	Trapèze pour n impair $\frac{\sqrt{3}}{2n}$	Trapèze pour n impair $\frac{\sqrt{3}}{n-1}$	Trapèze pour n impair $\sqrt{\frac{\sqrt{3}}{n-2}}$

TABLE 5.1 – Polygones convexes équilatéraux de périmètre, diamètre ou aire unitaire qui maximisent ou minimisent la largeur.

de maximiser l'aire à diamètre unitaire qui utilise des éléments de preuve purement géométriques, des concepts de théorie des graphes et des algorithmes d'optimisation globale robustes et exacts [5, 7, 9].

Dans un premier temps pour notre étude sur les polygones convexes équilatéraux, nous avons vérifié numériquement la véracité de notre intuition avant de commencer la démonstration du théorème 5.1.7. Nous nous sommes ensuite intéressés au cas de la maximisation de la somme des distances à largeur unitaire qui reste encore ouvert, afin de proposer une conjecture.

Considérons un polygone convexe équilatéral E_n à n côtés. Notons A_n son aire, P_n son périmètre, D_n son diamètre, W_n sa largeur et S_n la somme des distances entre toutes les paires de sommets. Notons $V_i = (x_i, y_i)$ les sommets consécutifs de E_n. Alors :

$$
\begin{aligned}
A_n &= |\frac{1}{2}\sum_{i=1}^{n}(y_{i+1} - y_i)(x_{i+1} + x_i)|,\\
P_n &= \sum_{i=1}^{n}\|V_{i+1} - V_i\|,\\
D_n &= \max_{i<j}\|V_i - V_j\|,\\
S_n &= \sum_{i<j}\|V_i - V_j\|,\\
W_n &= \min_{i}\max_{j\neq i,i+1}\frac{|(y_{i+1} - y_i)x_i + (x_j - x_{j+1})y_i + x_{j+1}y_j - x_j y_{j+1}|}{\|V_i - V_j\|},
\end{aligned}
$$

avec $i+1$ et $j+1$ pris modulo n.

Maximisation de l'aire à largeur unitaire

Le problème considéré consiste à maximiser l'aire d'un pentagone équilatéral de largeur unitaire. Il peut se formuler comme un problème d'optimisation globale de

la forme suivante :

$$\begin{cases} \max & A_n \\ s.t. & W_n = 1, \\ & \|V_2 - V_1\| = \|V_3 - V_2\| = \cdots = \|V_n - V_{n-1}\| = \|V_1 - V_n\|. \end{cases}$$

L'expression de la largeur n'est pas une quantité facile à manipuler en optimisation globale. Nous avons donc préféré reformuler le problème afin de faciliter la résolution.

Pour $n = 5$, sans perdre de généralité, V_5 est fixé à l'origine, V_4 sur la droite $y = 0$, x_1, x_2 et x_3 sont ordonnés et y_1, y_2 et y_3 sont compris dans $[0, 1]$ pour imposer implicitement la largeur à 1. De plus, d'après le théorème 5.1.3, la longueur des côtés est inférieure à $\frac{2}{\sqrt{3}}$. On peut donc limiter le domaine d'étude à : $x_1 \in [-\frac{2}{\sqrt{3}}, 0], x_2 \in [-\frac{2}{\sqrt{3}}, \frac{2}{\sqrt{3}}], x_3 \in [-\frac{2}{\sqrt{3}}, \frac{4}{\sqrt{3}}],\ x_4 \in [0, \frac{2}{\sqrt{3}}]$. Notons ce domaine $\mathbf{X}_5$.

Cette formulation permet de relâcher la problématique. Néanmoins, si la solution du problème suivant à une largeur unitaire, cela veut dire que la solution trouvée est l'optimum du problème original.

$$\begin{cases} \min\limits_{\substack{(x_1,x_2,x_3,x_4)\in\mathbf{X}_5,\\ (y_1,y_2,y_3)\in[0,1]^3}} & \frac{1}{2}\left(x_1y_1 + (y_2 - y_1)(x_2 + x_1) + (y_3 - y_2)(x_3 + x_2) - y_3(x_4 + x_3)\right) \\ s.t. & \\ & x_4^2 = x_1^2 + y_1^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2, \\ & x_4^2 = (x_2 - x_3)^2 + (y_2 - y_3)^2 = (x_3 - x_4)^2 + y_3^2, \\ & x_1 \leq x_2 \leq x_3. \end{cases}$$

Pour résoudre ce problème de manière robuste et fiable, afin de garantir le résultat, nous avons donc utilisé notre algorithme *IBBA* en incluant la propagation des contraintes (voir la section 1.3.2) et la technique de reformulation affine robuste (voir la section 3.2). L'algorithme résout le problème en 1.4 secondes et 668 itérations et la solution est bien le trapèze $\hat{T}_5$ comme le prouve le lemme 5.1.6. L'algorithme a néanmoins réussi à le prouver avec une erreur de seulement 10^{-10} sur les contraintes et sur l'encadrement de la valeur du minimum global.

Pour le cas $n = 7$, on procède de la même façon en fixant $V_7 = (0, 0)$ et V_6 sur la droite $y = 0$. Ainsi les variables peuvent être restreintes à : $x_1 \in [-\frac{2}{\sqrt{3}}, 0], x_2 \in [-\frac{4}{\sqrt{3}}, \frac{2}{\sqrt{3}}], x_3 \in [-\frac{6}{\sqrt{3}}, \frac{4}{\sqrt{3}}], x_4 \in [-\frac{4}{\sqrt{3}}, \frac{6}{\sqrt{3}}], x_5 \in [-\frac{2}{\sqrt{3}}, \frac{4}{\sqrt{3}}], x_6 \in [0, \frac{2}{\sqrt{3}}]$.
Notons ce domaine $\mathbf{X}_7$.

La formulation relâché du problème est :

$$\begin{cases} \max\limits_{\substack{(x_1,x_2,x_3,x_4,x_5,x_6)\in\mathbf{X}_7,\\ (y_1,y_2,y_3,y_4,y_5)\in[0,1]^5}} & A_7 \\ s.t. & \\ & x_6^2 = x_1^2 + y_1^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2, \\ & x_6^2 = (x_2 - x_3)^2 + (y_2 - y_3)^2 = (x_3 - x_4)^2 + (y_3 - y_4)^2, \\ & x_6^2 = (x_4 - x_5)^2 + (y_4 - y_5)^2 = (x_5 - x_6)^2 + y_5^2, \end{cases}$$

avec $A_7 = -\frac{1}{2}(x_1y_1 + (y_2 - y_1)(x_2 + x_1) + (y_3 - y_2)(x_3 + x_2) + (y_4 - y_3)(x_4 + x_3) + (y_5 - y_4)(x_5 + x_4) - y_5(x_6 + x_5))$.

Notre algorithme trouve le minimum global $\hat{T}_7$ avec une certification de 10^{-10} sur les contraintes et l'encadrement. Ce problème est résolu en 21.16 secondes et 4816 itérations.

Maximisation de la somme des distances à largeur unitaire

Le problème suivant consiste à maximiser la somme des distances d'un pentagone équilatéral de largeur unitaire. Il peut se formuler comme un problème d'optimisation globale de la forme suivante :

$$\left\{\begin{array}{ll}\max & S_n \\ s.t. & W_n = 1, \\ & \|V_2 - V_1\| = \|V_3 - V_2\| = \cdots = \|V_n - V_{n-1}\| = \|V_1 - V_n\|.\end{array}\right.$$

Pour ce problème, à notre connaissance, il n'existe pas de démonstration mathématique complète. Notre algorithme peut donc être très utile pour voir la forme de la solution avec une précision de 10^{-10}.

Pour $n = 5$, tout comme dans la section précédente, nous avons utilisé la même reformulation afin de simplifier l'expression du problème. On obtient :

$$\left\{\begin{array}{lll}\max\limits_{\substack{x_1,x_2,x_3,x_4\in\mathbf{X}_5,\\ y_1,y_2,y_3\in[0,1]^3}} & & S_5 \\ s.t. & x_4^2 = & x_1^2 + y_1^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2 \\ & = & (x_2 - x_3)^2 + (y_2 - y_3)^2 = (x_3 - x_4)^2 + y_3^2, \\ & & x_1 \leq x_2 \leq x_3,\end{array}\right.$$

avec $S_5 = 5x_4 + \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} + \sqrt{(x_1 - x_4)^2 + y_1^2} + \sqrt{(x_2 - x_4)^2 + y_2^2} + \sqrt{x_2^2 + y_2^2} + \sqrt{x_3^2 + y_3^2}$

Le problème est résolu de façon robuste et fiable en 1.5 secondes et 578 itérations avec un certificat à 10^{-10} près. La solution est le trapèze $\hat{T}_5$ avec $\hat{S}_5 = 14.3230484$.

Pour $n = 7$, on procède de la même façon que pour la section 5.1.5. Le problème s'écrit :

$$\left\{\begin{array}{lll}\max\limits_{\substack{(x_1,x_2,x_3,x_4,x_5,x_6)\in\mathbf{X}_7,\\ (y_1,y_2,y_3,y_4,y_5)\in[0,1]^5}} & & S_7 \\ s.t. & x_6^2 = & x_1^2 + y_1^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2, \\ & x_6^2 = & (x_2 - x_3)^2 + (y_2 - y_3)^2 = (x_3 - x_4)^2 + (y_3 - y_4)^2, \\ & x_6^2 = & (x_4 - x_5)^2 + (y_4 - y_5)^2 = (x_5 - x_6)^2 + y_5^2,\end{array}\right.$$

avec $S_7 = 7x_6 + \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} + \sqrt{(x_1 - x_4)^2 + (y_1 - y_4)^2} + \sqrt{(x_1 - x_5)^2 + (y_1 - y_5)^2} + \sqrt{(x_1 - x_6)^2 + y_1^2} + \sqrt{(x_2 - x_4)^2 + (y_2 - y_4)^2} + \sqrt{(x_2 - x_5)^2 + (y_2 - y_5)^2} + \sqrt{(x_2 - x_6)^2 + y_2^2} + \sqrt{x_2^2 + y_2^2} + \sqrt{(x_3 - x_5)^2 + (y_3 - y_5)^2} + \sqrt{(x_3 - x_6)^2 + y_3^2} + \sqrt{x_3^2 + y_3^2} + \sqrt{(x_4 - x_6)^2 + y_4^2} + \sqrt{x_4^2 + y_4^2} + \sqrt{x_5^2 + y_5^2}$.

La solution de ce problème est également le trapèze $\hat{T}_7$ avec $\hat{S}_7 = 37.2041169$ et notre algorithme le prouve en 25.08 secondes et 3384 itérations avec une certification à 10^{-10} près.

Pour $n = 9$, les équations se compliquent encore ainsi que le nombre de variable, mais le principe reste le même. Notons le domaine $\mathbf{X}_9$: $x_1 \in [-\frac{2}{\sqrt{3}}, 0], x_2 \in [-\frac{4}{\sqrt{3}}, \frac{2}{\sqrt{3}}], x_3 \in [-\frac{6}{\sqrt{3}}, \frac{4}{\sqrt{3}}], x_4 \in [-\frac{8}{\sqrt{3}}, \frac{6}{\sqrt{3}}], x_5 \in [-\frac{6}{\sqrt{3}}, \frac{8}{\sqrt{3}}], x_6 \in [-\frac{4}{\sqrt{3}}, \frac{6}{\sqrt{3}}], x_7 \in [-\frac{2}{\sqrt{3}}, \frac{4}{\sqrt{3}}]$ et $x_8 \in [0, \frac{2}{\sqrt{3}}]$.

$$\begin{cases} \max\limits_{\substack{(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8)\in\mathbf{X}_9,\\ (y_1,y_2,y_3,y_4,y_5,y_6,y_7)\in[0,1]^7}} & S_9 \\ s.t. & x_8^2 = x_1^2 + y_1^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2, \\ & x_8^2 = (x_2 - x_3)^2 + (y_2 - y_3)^2 = (x_3 - x_4)^2 + (y_3 - y_4)^2, \\ & x_8^2 = (x_4 - x_5)^2 + (y_4 - y_5)^2 = (x_5 - x_6)^2 + (y_5 - y_6)^2, \\ & x_8^2 = (x_6 - x_7)^2 + (y_6 - y_7)^2 = (x_7 - x_8)^2 + y_7^2, \end{cases}$$

avec $S_9 = 9x_8 + \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} + \sqrt{(x_1 - x_4)^2 + (y_1 - y_4)^2} +$
$\sqrt{(x_1 - x_5)^2 + (y_1 - y_5)^2} + \sqrt{(x_1 - x_6)^2 + (y_1 - y_6)^2} + \sqrt{(x_1 - x_7)^2 + (y_1 - y_7)^2}$
$+\sqrt{(x_1 - x_8)^2 + y_1^2} + \sqrt{(x_2 - x_4)^2 + (y_2 - y_4)^2} + \sqrt{(x_2 - x_5)^2 + (y_2 - y_5)^2} +$
$\sqrt{(x_2 - x_6)^2 + (y_2 - y_6)^2} + \sqrt{(x_2 - x_7)^2 + (y_2 - y_7)^2} + \sqrt{(x_2 - x_8)^2 + y_2^2} +$
$\sqrt{x_2^2 + y_2^2} + \sqrt{(x_3 - x_5)^2 + (y_3 - y_5)^2} + \sqrt{(x_3 - x_6)^2 + (y_3 - y_6)^2} +$
$\sqrt{(x_3 - x_7)^2 + (y_3 - y_7)^2} + \sqrt{(x_3 - x_8)^2 + y_3^2} + \sqrt{x_3^2 + y_3^2} + \sqrt{(x_4 - x_6)^2 + (y_4 - y_6)^2}$
$+\sqrt{(x_4 - x_7)^2 + (y_4 - y_7)^2} + \sqrt{(x_4 - x_8)^2 + y_4^2} + \sqrt{x_4^2 + y_4^2} +$
$\sqrt{(x_5 - x_7)^2 + (y_5 - y_7)^2} + \sqrt{(x_5 - x_8)^2 + y_5^2} + \sqrt{x_5^2 + y_5^2} + \sqrt{(x_6 - x_8)^2 + y_6^2} +$
$\sqrt{x_6^2 + y_6^2} + \sqrt{x_7^2 + y_7^2}.$

L'algorithme prouve à 10^{-10} près que le trapèze $\hat{T}_9$ est la solution optimale de ce problème avec $\hat{S}_9 = 76.4254863$. La solution est obtenue en 34 minutes 17 secondes et 82 155 itérations.

Remarque 5.1.8 *En regardant les 3 problèmes précédents, on peut remarquer la très nette croissance exponentielle du temps de calcul. Ceci est dû à la nature même de l'algorithme dont la complexité dépend fortement du nombre de variables. Pour ce type de problème, un algorithme spécialisé dans les problèmes quadratiques non convexes, tel que [8], aurait été sûrement mieux adapté pour les résoudre. Néanmoins, la nature non convexe des problèmes implique dans une majorité des cas que les problèmes seront compliqués à résoudre, même s'ils ne sont que quadratiques. De plus, l'avantage de notre algorithme est de certifier de manière fiable et robuste le résultat obtenu.*

Remarque 5.1.9 *La résolution de ces problèmes a aussi été effectuée sans la technique rART et uniquement avec la propagation des contraintes. En moyenne, le nombre d'itérations était multiplié par* 100 *ainsi que l'espace mémoire nécessaire. Ceci montre encore l'intérêt de cette technique de reformulation rART.*

Suite aux résultats obtenus numériquement, il est tout à fait réaliste de proposer la conjecture suivante :

Conjecture 5.1.10 *La somme des distances entre toutes les paires de sommets de tout polygone convexe équilatéral à* $(2m+1)$ *côtés et de largeur unitaire est majorée et atteinte par celle du trapèze* $\hat{T}_n$.

5.1.6 Discussion

Dans cette section, trois problèmes ouverts sont démontrés sur les polygones convexes équilatéraux de largeur unitaire :

- maximiser le périmètre : théorème 5.1.3,
- maximiser le diamètre : corollaire 5.1.4,
- maximiser l'aire : théorème 5.1.7

Un dernier problème, celui de la maximisation de la somme des distances, a été conjecturé et démontré numériquement pour $n = 5$, 7 et 9.

Pour tous ces problèmes, la solution est un polygone convexe équilatéral de largeur unitaire et arbitrairement proche du trapèze $\hat{T}_n$ de la figure 5.1.

5.2 Résolution d'un problème inverse de dimensionnement de moteurs électriques

En génie électrique, les problèmes de dimensionnement sont à la fois difficiles à résoudre et aussi difficiles à modéliser. Cela consiste à résoudre dans la plupart des cas un *problème inverse*, c'est-à-dire par exemple pour un moteur électrique on impose le couple du moteur, la densité de flux maximal dans le cuivre, le champ magnétique maximal, etc. et à partir de ces caractéristiques, on essaie de trouver les dimensions physiques du moteur qui répond aux conditions souhaitées. Or dans la majorité des problèmes inverses, nous ne disposons que de codes d'éléments finis qui obligent l'utilisation d'algorithmes gérant des fonctions de type boîte noire, tel que des algorithmes génétiques, des algorithmes Derivative-Free Optimization [24] ou de Direct Search [1, 6]. Malheureusement, ces algorithmes n'apportent dans le meilleur des cas qu'une information locale sur la qualité de la solution retournée.

L'intérêt majeur de notre approche est donc de certifier que la solution retournée est bien le minimum global. Néanmoins, un travail préalable est nécessaire pour modéliser et obtenir une formulation explicite du problème. De plus, une fois les équations explicites obtenues, les problèmes restent encore très compliqués à résoudre avec une infinité de minima locaux, ce qui nous oblige à le considérer comme un problème d'optimisation globale, car la différence entre une solution locale et globale est souvent loin d'être négligeable.

Le problème de dimensionnement que nous allons chercher à résoudre est issu du travail de l'équipe GREM3 du laboratoire LAPLACE de Toulouse. Cette équipe dirigée par Nogarède effectue des recherches sur la modélisation des moteurs électriques [33, 35, 36], des coupleurs magnétiques [34] et des actionneurs piézo-électriques [89]. Ils ont notamment réussi à extraire des fonctions explicites pour modéliser les champs magnétiques, les forces de couplage, etc.

Grâce à une interaction très fructueuse entre Nogarède et Messine [68, 73, 74], la découverte de solutions globales des problèmes existants a permis de repousser toujours plus loin la complexité des modèles traités, jusqu'à obtenir celui que nous avons abordé dans cette section.

5.2.1 Modélisation

La modélisation du problème est issue des articles de Fitan, Messine et Nogarede [32, 71]. Il s'agit du problème de dimensionnement d'un moteur électrique formulé comme un problème mixte d'optimisation globale avec contraintes.

Les variables

Pour faciliter la compréhension mathématique du problème, nous avons classé les variables du modèles en quatre catégories et en les renommant :

– Les variables continues notées x_i : elles représentent les dimensions physiques du moteur, des angles, des pourcentages ou des températures ;

- Les variables booléennes notées b_i : elles permettent d'activer et de désactiver certaine partie du modèle pour pouvoir considérer en même temps plusieurs types de moteur ;
- Les variables entières notées z_i : elles représentent les quantités dénombrables du problème, tel que le nombre de paires de pôles, le nombre de phases, le nombre d'encoches.
- Les variables de catégories notées k_i : elles représentent les différentes caractéristiques des types de matériaux qu'il est possible d'utiliser pour construire le moteur. Leurs valeurs appartiennent à un ensemble fini.

Dans le tableau 5.2, nous avons répertorié toutes les variables. La première colonne représente son nom dans notre formulation, la deuxième colonne représente la lettre généralement utilisée dans la formulation écrite par les chercheurs de l'équipe GREM3 et la dernière colonne explique la signification physique de la variable.

Les contraintes

En considérant le problème de dimensionnement du moteur électrique, il apparaît plusieurs types de contraintes : (i) les contraintes magnétiques, (ii) les contraintes thermiques, (iii) les contraintes physiques.

(i) Parmi les contraintes magnétiques, la plus importante est celle concernant le couple magnétique Γ_{em}. Celui-ci est directement lié à la puissance du moteur, par contre c'est aussi la contrainte qui est la plus dure à calculer, car elle est liée à de nombreuses autres grandeurs physiques.

- La charge linéique statorique K_S est bornée par les caractéristiques physiques du cuivre :
$$K_S(x,z,b,k) = k_r x_5 x_8 \left(b_2 k_d(x,z,b,k) + (1-b_2)\right),$$
avec k_r le coefficient de remplissage des bobinages.
- Le coefficient du couple k_Γ est obtenu par la formule suivante :
$$k_\Gamma(x,z,b,k) = \frac{\pi}{2}\left[b_3\left(1-K_f(x,z,b,k)\right)\sqrt{x_7} + (1-b_3)\frac{\sqrt{2}}{2}\sin\left(x_7\frac{\pi}{2}\right)\right].$$
- L'équation du coefficient de fuites semi-empiriques K_f est issue d'observations expérimentales et de simulations numériques. Elle est bornée par des valeurs empiriques correspondant à des caractéristiques physiques du moteur :
$$K_f(x,z,b,k) = (1-b_2)b_3 \times \left(1.5 z_1 x_7 \left[\frac{x_5+x_3}{x_1}\right]\right).$$
- La valeur de l'induction à vide B_e est bornée pour que les valeurs restent en accord avec les hypothèses de la modélisation :
$$B_e(x,z,b,k) = \frac{2J(k_1)x_4}{(2b-1)k_c(x,z,b,k)x_1 \ln\left(\dfrac{x_1+2(2b_1-1)(1-b_2)x_5}{x_1-2(2b_1-1)(x_3+x_4)}\right)},$$
avec $J(k_1)$ le vecteur de polarisation magnétique du type d'aimant utilisé.
- Le coefficient de Carter k_c traduit l'effet des encoche sur l'induction à vide :

TABLE 5.2 – Détails des variables utilisées

Variable	Lettre	Description
x_1	D	Le diamètre d'alésage, i.e le diamètre intérieur du moteur
x_2	L	La longueur du moteur
x_3	e	L'espace de l'entrefer, i.e l'espace entre la partie statique et celle en mouvement
x_4	l_a	L'épaisseur des aimants
x_5	E	L'épaisseur des encoches
x_6	C	L'épaisseur de la culasse
x_7	β	Le % de l'arc polaire de l'aimant, i.e la largeur de l'aimant
x_8	j	La densité de courant dans les bobinages
x_9	a	La largeur d'une encoche
x_{10}	d	La largeur d'une dent
x_{11}	Θ_J	La température de la partie extérieure du moteur
x_{12}	Θ_c	La température dans l'entrefer du moteur
x_{13}	Θ_e	La température ambiante du milieu
z_1	p	Le nombre de paires de pôles
z_2	q	Le nombre de phases par paire de pôles
z_3	m	Le nombre d'encoches par phase par paire de pôles
b_1	b_r	Si $b_1 = 1$, la partie rotative du moteur est à l'intérieur, sinon, c'est la partie externe qui est rotative
b_2	b_e	Si $b_2 = 1$, le moteur a des encoches, sinon, il n'en a pas
b_3	b_f	Si $b_3 = 1$, l'alimentation est en créneaux rectangulaires, sinon, elle est sinusoïdale
k_1	σ_m	Le matériau utilisé pour les aimants
k_2	σ_{mt}	Le matériau magnétique utilisé pour les dents et la culasse

$$k_c(x, z, b, k) = \frac{1}{1 - b_2 \left(\dfrac{N_e(x, z, b, k) x_9^2}{5\pi x_1 x_3 + \pi x_1 x_9} \right)}.$$

– Le nombre total d'encoche N_e peut être retrouvé de deux façons différentes dans notre modèle. Cela permet de contraindre les variables par l'égalité suivante :

$$N_e(x, z, b, k) = b_2 \times \pi \left(\frac{x_1 + (2b_1 - 1)x_5}{x_9 + x_{10}} \right) = b_2 \times 2 z_1 z_2 z_3.$$

– Les champs magnétiques dans la culasse B_c et dans les dents B_t sont bornés par la valeur du champ de saturation $B_M(k_2)$ imposé par le matériau utilisé k_2 pour les dents et la culasse :

$$B_t(x,z,b,k) = b_2\frac{x_9+x_{10}}{x_{10}}B_e(x,z,b,k),$$
$$B_c(x,z,b,k) = \frac{x_1}{2z_1x_6}\left(\frac{\pi x_7}{2}(1-b_3)+b_3\right)B_e(x,z,b,k).$$

– Dans le cas d'une machine à encoches, on ajoute un facteur k_d égal au rapport de la largeur des dents et du pas d'encochage :

$$k_d(x,z,b,k) = b_2\frac{x_9}{x_9+x_{10}}.$$

– Le couple magnétique Γ_{em} est généralement contraint entre 10 et 20 Nm. Comme on peut le voir dans l'équation suivante, sa valeur est liée à toutes les caractéristiques définies précédemment :

$$\Gamma_{em}(x,z,b,k) = x_1x_2\left(x_1+(1-b_2)(2b_1-1)x_5\right) \times k_\Gamma(x,z,b,k)B_e(x,z,b,k)K_S(x,z,b,k).$$

(ii) Les contraintes thermiques sont imposées pour éviter que la température ne grimpe trop à l'intérieur du moteur. Cette contrainte est un modèle simplifié correspondant au calcul du flux thermique à l'intérieur du moteur :

$$\phi_{Th}(x,z,b,k) = \rho_{co}\pi x_8x_2\left(x_1+(2b_1-1)(1-b_2)x_5\right)\times K_S(x,z,b,k),$$
$$\phi_{Th}(x,z,b,k) = \pi x_2\left(h_a\left(x_1+2(2b_1-1)(x_5+x_6)\right)(x_{11}-x_{12})+x_1h_e(x_{11}-x_{13})\right),$$

avec ρ_{co} la résistivité du cuivre et h_a et h_e les coefficients de convection respectivement avec l'air ambiant et avec l'air de l'entrefer.

(iii) Des contraintes physiques sont imposées pour éviter que des moteurs trop plats ou trop longs ne soient trouvés, car ils sont généralement irréalisables. Pour éviter cela, on fixe donc un rayon interne minimal et un rayon externe maximal :

$$R_{int} = \frac{x_1}{2} - x_6 - (1-b_1)x_5 - b_1(x_3+x_4),$$
$$R_{ext} = \frac{x_1}{2} + x_6 + b_1x_5 + (1-b_1)(x_3+x_4).$$

Les objectifs

Pour ce problème, plusieurs fonctions objectifs peuvent être considérées, elles dépendent surtout de l'objectif industriel fixé. Il est possible de minimiser la masse totale du moteur M_a, le volume total V_g, ou encore uniquement le volume des aimants V_m qui représente la partie la plus coûteuse du moteur, etc.

Les formulations des fonctions objectifs s'écrivent comme suit :

– le volume des aimants V_m :

$$V_m(x,z,b,k) = \pi x_7x_2x_4\left(x_1-(2b_1-1)(2x_3+x_4)\right),$$

– le volume de la culasse V_c :

$$V_c(x,z,b,k) = 2\pi x_2x_6\left(x_1+(2b_1-1)(x_5-x_3-x_4)\right),$$

– le volume des dents V_t :

$$V_t(x,z,b,k) = \pi x_2x_5(x_1+(2b_1-1)x_5)\left(b_2(1-x_7)+(1-b_2)\left(\frac{x_{10}}{x_9+x_{10}}\right)\right),$$

– le volume du cuivre V_{co} :

$$V_{co}(x,z,b,k) = k_r \pi x_2 x_5 (x_1 + (2b_1 - 1)x_5) \left((1-b_2)x_7 + b_2 \left(\frac{x_9}{x_9 + x_{10}} \right) \right),$$

– le volume global du moteur V_g :

$$V_g(x,z,b,k) = \frac{\pi x_2}{4} \left(b_1 \left(x_1 + 2(x_5 + x_6) \right)^2 + (1-b_1) \left(x_1 + 2(x_3 + x_4 + x_6) \right)^2 \right),$$

– la masse totale du moteur M_a :

$$\begin{aligned} M_a(x,z,b,k) = {} & d_{PM}(k_1) V_m(x,z,b,k) + d_{CM}(k_2) V_c(x,z,b,k) \\ & + \left(d_{CM}(k_2) b_2 + d_{Al}(1-b_2) \right) V_t(x,z,b,k) \\ & + d_{co} V_{co}(x,z,b,k), \end{aligned}$$

avec d_{PM} la densité du matériau utilisé pour les aimants, d_{CM} la densité du matériau utilisé pour la culasse, d_{Al} la densité de l'aluminium et d_{co} la densité du cuivre.

Les paramètres

En physique, la différence entre une variable et un paramètre n'est pas exactement la même que celle définie en optimisation. En optimisation, les variables sont les quantités que l'algorithme va faire varier pour trouver l'optimum, et les paramètres sont les quantités fixées au début de l'optimisation. En physique, une variable est associée à une caractéristique qui peut varier, contrairement à un paramètre qui a une valeur immuable et indépendante du problème. C'est pourquoi pendant la résolution certaines variables seront fixées.

Les paramètres physiques sont donc les constantes suivantes :

$$\begin{aligned} k_r &= 0.7, & \rho_{co} &= 0.18\ 10^{-6}\ \Omega m, \\ h_a &= 4, & d_{Al} &= 2700\ kg/m^3, \\ h_e &= 12, & d_{co} &= 8920\ kg/m^3. \end{aligned}$$

En ce qui concerne les variables de catégorie k_1 et k_2, nous n'avons que deux catégories par matériau dans cette étude, nous les considérerons donc par la suite comme des variables binaires. Voici le détail des paramètres associés aux variables k_1 et k_2 :

Variable	Valeur	Matériau	J	B_M	d_{PM}	d_{CM}
k_1	0	Moderne NdFeB	$0.9T$	-	$7900kg/m^3$	-
	1	Plasto-Aimant	$0.6T$	-	$6000kg/m^3$	-
k_2	0	Tôles	-	$1.5T$	-	$7900kg/m^3$
	1	Poudre	-	$1.2T$	-	$6000kg/m^3$

La formulation générale

Le problème général s'exprime sous la forme d'un problème d'optimisation globale de type (MINLP). L'équation (PB_m) reprend le détail des contraintes précédentes, avec f une fonction objectif au choix parmi celles proposées.

$$\left\{\begin{array}{ll} \min\limits_{\substack{(x\in X\subset\mathbb{R}^{13},z\in Z\subseteq\mathbb{N}^3\\ b\in B\subseteq\mathcal{B}^3,k\in K\subseteq\mathcal{B}^2}} & f(x,z,b,k) \\ s.t. & \Gamma_{em}(x,z,b,k)=10, \\ & 0.4\leq B_e(x,z,b,k)\leq 0.9, \\ & 500.\leq K_S(x,z,b,k)\leq 12000, \\ & 0.1(1-b_2)\leq K_f(x,z,b,k)\leq 0.3, \\ & 0.4b_2\leq k_d(x,z,b,k)\leq 0.6, \\ & 0.9\leq B_c(x,z,b,k)\leq B_M(k_2), \\ & 0.9b_2\leq B_t(x,z,b,k)\leq B_M(k_2), \\ & R_{int}(x,z,b,k)\geq 50.10^{-3}, \\ & R_{ext}(x,z,b,k)\leq 80.10^{-3}, \\ & \text{Contrainte d'égalité thermique } \phi_{Th}, \\ & \text{Contrainte d'égalité sur le nombre d'encoche } N_e. \end{array}\right. \quad (PB_m)$$

5.2.2 Résolution du problème d'optimisation globale

Dans cette étude, nous avons axé notre recherche sur les moteurs ayant un couple magnétique de $10Nm$. Nous avons donc réduit notre espace de recherche pour tenir compte uniquement des moteurs *physiquement réalisables*. Cet espace de recherche, noté X_{moteur}, est donc :

$$\begin{array}{c} X=[1.10^{-3},0.3]\times[0.050,0.150]\times[0.001]\times[0.004,0.1]\times[0.004,0.1]\times[0.004,0.1] \\ \times[0.85]\times[3000000,6000000]\times[0.004,0.1]\times[0.004,0.1]\times[130]\times[55]\times[50], \end{array}$$

$$Z=\ [1,16]\times[3]\times[1,2],\quad B=\ [0,1]\times[0,1]\times[1],\quad K=\ [0,1]\times[0,1],$$

$$X_{moteur}=X\times Z\times B\times K.$$

Contrairement aux problèmes de géométrie plane du chapitre 5.1, les contraintes de ce problème ne sont pas des contraintes nécessitant une vérification très précise. Des tolérances plus larges ont donc été imposées sur les contraintes. Le tableau 5.3 résume les tolérances choisies :

La résolution du problème (PB_m) a été effectuée avec l'algorithme *IBBA* (algorithme 2) munie de la technique de propagation de contraintes (algorihtme 3) en y ajoutant ou non les techniques d'accélération développées dans le chapitre 3 : (i) ART (alogrithme 8), (ii) rART (algorithme 9) et (iii) MART (section 3.3 et algorithme 8).

Contraintes	Tolérances
$\Gamma_{em}(x,z,b,k) = 10$	10^{-2}
$0.4 \leq B_e(x,z,b,k) \leq 0.9$	10^{-3}
$500. \leq K_S(x,z,b,k) \leq 12000.$	10^{0}
$0.1(1-b_2) \leq K_f(x,z,b,k) \leq 0.3$	10^{-4}
$0.4b_2 \leq k_d(x,z,b,k) \leq 0.6$	10^{-4}
$0.9 \leq B_c(x,z,b,k) \leq B_M(k_2)$	10^{-4}
$0.9b_2 \leq B_t(x,z,b,k) \leq B_M(k_2)$	10^{-4}
$R_{int}(x,z,b,k) \geq 50.10^{-3}$	10^{-4}
$R_{ext}(x,z,b,k) \leq 80.10^{-3}$	10^{-4}
Contrainte d'égalité thermique ϕ_{Th}	10^{-1}
Contrainte d'égalité sur N_e	10^{-4}

TABLE 5.3 – Tableau des tolérances accordées sur les contraintes de (PB_m).

Nous avons considéré trois fonctions objectifs : (a) la masse totale M_a (tableau 5.4), (b) le volume global V_g (tableau 5.5) et (c) une combinaison pondérée des deux objectifs précédents (tableau 5.6), dont voici l'équation :

$$MULTI(x,z,b,k) = \frac{V_g(x,z,b,k)}{9.28\ 10^{-4}} + \frac{M_a(x,z,b,k)}{2.24}. \tag{5.2}$$

La précision demandée sur la fonction objectif est de 10^{-3} pour la masse et de 10^{-6} pour le volume et la combinaison pondérée.

Le code est écrit en Fortran 90/95 et utilise le compilateur *f90* de SUN, avec la librairie incluse permettant d'utiliser l'arithmétique d'intervalles. Pour résoudre le programme linéaire généré par notre technique, on utilise C-PLEX version 11.0. Tous les tests sont réalisés sur un PC-Intel-Xeon-3Gz avec 2Go de RAM et un système Linux 64 bit. Nous avons ajouté un test d'arrêt uniquement sur le nombre d'éléments maximal pouvant être stocké dans $\mathcal{L}$. L'algorithme s'arrête donc si la taille de $\mathcal{L}$ atteint 2 000 000, pour éviter les swaps sur le disque dur.

Les tableaux 5.4, 5.5 et 5.6 résument les résultats obtenus. La colonne "Algorithme" indique l'algorithme muni de la technique de reformulation qui a été utilisée. Les colonnes "ok ?", "t", "iter" et "max|$\mathcal{L}$|" désigne respectivement si la résolution a abouti, le temps de calcul, le nombre d'itérations et la taille maximale de la structure de donnée $\mathcal{L}$. La dernière colonne indique la valeur du critère de la solution trouvée. Les détails des solutions obtenues peuvent être consultés en annexe B.

Algorithme	ok ?	t	iter	max$\|\mathcal{L}\|$	M_a (kg)
IBBA	F	20min	5122950	2000000	2.317710
IBBA + ART	T	34min	647735	237809	2.244968
IBBA + rART	T	4h 12min	347740	159460	2.244968
IBBA + MART	T	1h 31min	1200876	260329	2.245643

TABLE 5.4 – Résultats de la minimisation de la masse M_a de (PB_m).

Algorithme	ok ?	t	iter	max$\|\mathcal{L}\|$	V_g ($.10^{-4}m^3$)
IBBA	F	53min	17413329	2000000	-
IBBA + ART	T	1h 19min	1952410	534291	9.291141
IBBA + rART	T	10h 08min	1051000	328046	9.289738
IBBA + MART	T	3h 34min	4331147	729615	9.290403

TABLE 5.5 – Résultats de la minimisation du volume V_g de (PB_m).

En observant ces résultats, on peut en déduire :

- L'algorithme *IBBA* seul ne parvient pas à résoudre complètement les problèmes. Cela est dû notamment à la limite imposée sur la mémoire disponible. Une solution réalisable n'est trouvée que lors de la minimisation de la masse.
- L'algorithme le plus rapide est celui utilisant la technique ART. Il résout tous les problèmes en environ **3** fois moins de temps qu'avec la technique MART et environ **8** fois plus rapidement qu'avec la technique rART.
- Le résultat avec la technique rART est techniquement le plus fiable. Toutes les solutions obtenues avec les autres techniques sont très proches. En effet, en observant le détail des solutions de l'annexe B, l'algorithme a convergé avec toutes les techniques dans le même voisinage. Une méthodologie de résolution pourrait consister à résoudre une première fois le problème avec l'algorithme *IBBA+ART*, puis de lancer l'algorithme *IBBA+rART*, en lui donnant dès le début la solution obtenue par la précédente optimisation afin de la certifier numériquement. Une autre idée serait d'utiliser *fAF2* à la place de *rAF2* pour plus d'efficacité.
- En comparant les performances de la technique ART et MART, on s'aperçoit que les reformulations linéaires sont plus efficaces que les reformulations linéaires mixtes. Néanmoins, la technique MART peut être facilement améliorée au niveau de la résolution du programme linéaire mixte, car une résolution complète n'est pas nécessaire, seul un bon minorant est utile. Il pourrait donc être intéressant de n'effectuer que quelques itérations dans la résolution du problème linéaire mixte. Un autre effort pourrait être apporté en modifiant la technique de coupe de notre algorithme de Branch and Bound. En effet, actuellement la technique consiste à couper en priorité sur les variables entières, ce qui n'est plus nécessaire en utilisant la technique MART.

Algorithme	ok ?	t	iter	max$\|\mathcal{L}\|$	$MULTI$
IBBA	F	28min	7570755	2000000	-
IBBA + ART	T	40min	840621	357638	2.047986
IBBA + rART	T	5h 35min	448101	211471	2.047970
IBBA + MART	T	2h 04min	1528822	439097	2.047486

TABLE 5.6 – Résultats de la minimisation du critère $MULTI$ de (PB_m).

5.2.3 Discussion

La résolution de ce type de problème est déjà en soi une avancée, mais l'accélération de la résolution de ce type de problème va permettre d'ouvrir la voie aux chercheurs pour proposer des modèles encore plus complets avec des équations toujours plus complexes et négligeant de moins en moins de phénomènes électrodynamiques. De nombreux cas ne devraient plus rester hors de portée par notre approche algorithmique.

Conclusion

Jusqu'à dans les années 90, pour résoudre de manière efficace des problèmes d'optimisation globale, l'utilisateur devait toujours choisir entre des algorithmes de recherche locale, telle que la méthode du lagrangien augmenté, des algorithmes métaheuristiques basés sur une méthode de recherche locale, tels que les méthodes tabou [37] ou VNS [44], ou des techniques d'optimisation stochastique, telles que les algorithmes génétiques.

Les algorithmes d'optimisation globale déterministe ont fait d'énormes avancées ces dernières années en se révélant toujours plus efficaces pour résoudre de manière exacte un grand nombre de problèmes. Le logiciel le plus répandu est le logiciel open source *GlobSol*, écrit par Kearfott et son équipe [52], et basé sur un algorithme de Branch and Bound par intervalles, tout comme notre algorithme *IBBA*. Cependant, *IBBA* (Interval Branch and Bound Algorithm) a aussi montré son efficacité pour résoudre des problèmes industriels tels que, par exemple, le dimensionnement de machines électriques [32, 36, 71, 73], et grâce aux techniques développées dans cette étude, les limites de ce type d'algorithme ont pu encore être repoussées.

Les combinaisons IA_AF développées dans le deuxième chapitre sont prometteuses, car les encadrements obtenus sont toujours meilleurs que les autres arithmétiques. Mais, une étude plus approfondie doit être réalisée pour étudier ces avantages une fois insérés dans un algorithme d'optimisation globale. La méthode d'implémentation de la robustesse de la *floating-point affine arithmetic fAF2* donne des résultats nettement plus performants que les deux autres propositions, avec des temps de calcul **20** fois inférieures et des résultats équivalents à 10^{-10} près. Ainsi, la minimisation du nombre de changements du mode d'arrondi du processeur doit être absolument recherchée.

Dans le troisième chapitre, nous avons présenté un nouveau type de technique de relaxation linéaire fiable et robuste basée sur les formes affines et leurs arithmétiques. Ces méthodes construisent de manière automatique une relaxation linéaire de taille équivalente au problème original. Elle est très peu consommatrice en temps de calcul et permet d'améliorer les minorants en prenant en compte le système des contraintes relâchées dans leur calcul. L'efficacité de cette technique a été validée sur 74 problèmes tests issus de la librairie COCONUT, en divisant les temps de résolution par **10** en moyenne. Par ailleurs, l'utilisation de la technique de reformulation robuste avec l'arithmétique affine robuste fAF2 permet d'obtenir des temps équivalents aux versions non parfaitement robustes.

Toutefois, l'extension de cette technique au cas mixte nécessite encore des améliorations. Le programme linéaire mixte devrait être davantage paramétré pour accélérer la résolution complète de la méthode. Et une étude sur le choix des variables à privilégier pour découper durant le Branch and Bound devrait permettre d'améliorer les premiers résultats obtenus.

Dans le quatrième chapitre, nous présentons une méthodologie métaheuristique basée sur la limitation de la mémoire disponible d'un algorithme de Branch and Bound par intervalles. Une étude théorique sur la complexité en temps d'une telle approche lorsque la structure de données triée en largeur d'abord a abouti au résultat suivant : l'ordre de grandeur de la complexité en temps est polynomiale. Cette approche métaheuristique est validée en considérant une heuristique qui y est entièrement détaillée et qui permet de certifier tout de même un encadrement de la valeur du minimum global à la fin de l'exécution.

Dans la première section du dernier chapitre, nous démontrons que la solution de trois problèmes ouverts sur les polygones convexes équilatéraux de largeurs unitaires est arbitrairement proche du trapèze $\hat{T}_n$ de la figure 5.1 : il maximise le périmètre, le diamètre et l'aire. Le problème de la maximisation de la somme des distances est conjecturé en utilisant les preuves numériques obtenues pour les cas $n = 5$, 7 et 9, grâce à *IBBA*.

Dans la deuxième section du dernier chapitre, le problème de dimensionnement d'un moteur électrique est entièrement détaillé. Ce travail a pour objectif de faciliter la compréhension des modèles par les mathématiciens, pour permettre de plus grandes interactions entre les disciplines dont les électrotechniciens sont très demandeurs. L'utilisation des techniques décrites dans les chapitres précédents a permis de résoudre le problème de dimensionnement avec des temps de calcul très acceptables, compte tenu de la complexité des équations.

En optimisation globale, de nombreuses voies restent encore à explorer. Avec l'augmentation constante de la complexité des problèmes à traiter, la limite de notre algorithme doit sans cesse être repoussée. La complexité des équations ne représente pas un obstacle pour notre algorithme, comme nous l'avons vu avec la résolution du dimensionnement du moteur électrique. La taille des problèmes d'optimisation résolus de manière exacte dans cette thèse est conséquente pour ce type de problèmes. Jusqu'à présent, l'algorithme résout de façon efficace les problèmes ayant 30 variables et 50 contraintes au plus. En combinant l'ensemble des études réalisées au sein de cette thèse, la taille des problèmes à traiter pourra certainement encore être augmentée. Entre le début et la fin de notre étude, l'ordre de grandeur du temps de calcul de notre algorithme est passé de quelques jours à quelques heures pour les problèmes les plus complexes.

Bibliographie

[1] M. A. Abramson, C. Audet, J. E. Dennis, Jr. et S. Le Digabel : Orthomads : A deterministic mads instance with orthogonal directions. *SIAM Journal on Optimization*, 20(2):948–966, 2009.

[2] American National Standards Institute and Institute of Electrical and Electronic Engineers, éditeur. *IEEE standard for binary floating-point arithmetic*. Std 754-2008. ANSI/IEEE Standard, 2008.

[3] American National Standards Institute and Institute of Electrical and Electronic Engineers, éditeur. *IEEE Interval Standard Working Group - P1788*. ANSI/IEEE Standard, 2010.

[4] I.P. Androulakis, C.D. Maranas et C.A. Floudas : alpha BB : A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4):337–363, Jan 1995.

[5] C. Audet : *Optimisation globale structurée : propriétés, équivalences et résolution*. Thèse de doctorat, École Polytechnique de Montréal, 1997.

[6] C. Audet, J. E. Dennis, Jr. et S. Le Digabel : Globalization strategies for mesh adaptive direct search. *Computational Optimization and Applications*, 46(2):193–215, 2010.

[7] C. Audet, P. Hansen et S. Le Digabel : Exact solution of three nonconvex quadratic programming problems. *In Frontiers in Global Optimization*. Kluwer Academis Publishers, 2004.

[8] C. Audet, P. Hansen, B. Jaumard et G. Savard : A branch and cut algorithm for nonconvex quadratically constrained quadratic programming. *Mathematical Programming Series A*, 87(1):131–152, 2000.

[9] C. Audet, P. Hansen et F. Messine : Extremal problems for convex polygons. *Journal of Global Optimization*, 38(2):163–179, 2007.

[10] C. Audet, P. Hansen et F. Messine : The small octagon with longest perimeter. *Journal of Combinatorial Theory, Series A*, 114(1):135–150, Jan 2007.

[11] C. Audet, P. Hansen et F. Messine : Extremal problems for convex polygons - an update. *In Lectures on Global Optimization*, volume 55 de *Fields Institute Communications*, pages 1–16. American Mathematical Society, 2009.

[12] C. Audet, P. Hansen et F. Messine : Isoperimetric polygons of maximum width. *Discrete and Computational Geometry*, 41(1):45–60, 2009.

[13] C. Audet, P. Hansen, S. Perron et F. Messine : The minimum diameter octagon with unit-length sides : Vincze's wife's octagon is suboptimal. *Journal of Combinatorial Theory. Series A*, 108(1):63–75, 2004.

[14] C. Audet et J. Ninin : Maximal perimeter, diameter and area of equilateral unit-width convex polygons. Rapport technique, GERAD, en soumission, 2010.

[15] E. Baumann : Optimal centered forms. *BIT Numerical Mathematics*, 28(1):80–87, 1988.

[16] P. Belotti, S. Cafieri, J. Lee et L. Liberti : Feasibility-based bounds tightening via fixed points. *In Proceedings of COCOA'10*, Lecture Notes in Computer Science. à paraître, 2010.

[17] P. Belotti, J. Lee, L. Liberti, F. Margot et A. Waechter : Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods & Software*, 24(4-5):597–634, 2009.

[18] F. Benhamou, F. Goualard, L. Granvilliers et J.-F. Puget : Revising hull and box consistency. *In Proceedings of the 1999 international conference on Logic programming*, pages 230–244, Cambridge, 1999. The MIT Press.

[19] A. Bezdek et F. Fodor : On convex polygons of maximal width. *Archiv der Mathematik*, 74(1):75–80, 2000.

[20] E. Carrizosa, P. Hansen et F. Messine : Improving interval analysis bounds by translations. *Journal of Global Optimization*, 29(2):157–172, 2004.

[21] E. Carrizosa et F. Messine : An exact global optimization method for deriving weights from pairwise comparison matrices. *Journal of Global Optimization*, 38(2): 237–247, 2007.

[22] L.G. Casado, J.A. Martinez et I. García : Experiments with a new selection criterion in a fast interval optimization algorithm. *Journal of Global Optimization*, 19(3):247–264, 2001.

[23] J.L.D. Comba et J. Stolfi : Affine arithmetic and its applications to computer graphics. *In Proceedings of SIBGRAPI'93 - VI Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens*, pages 9–18, 1993.

[24] A.R. Conn, K. Scheinberg et L.N. Vicente : *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2009.

[25] A.E. Csallner, T. Csendes et M.C. Markot : Multisection in interval branch-and-bound methods for global optimization - I. Theoretical results. *Journal of Global Optimization*, 16(4):371–392, 2000.

[26] T. Csendes : Generalized subinterval selection criteria for interval global optimization. *Numerical Algorithms*, 37(1-4):93–100, 2004.

[27] T. Csendes et D. Ratz : Subdivision direction selection in interval methods for global optimization. *SIAM Journal on Numerical Analysis*, 34(3):922–938, 1997.

[28] B. Datta : A discrete isoperimetric problem. *Geometriae Dedicata*, 64(1):55–68, 1997.

[29] L. de Figueiredo et J. Stolfi : Adaptive enumeration of implicit surfaces with affine arithmetic. *Computer Graphics Forum*, 15(5):287–296, 1996.

[30] L. de Figueiredo et J. Stolfi : Affine arithmetic : Concepts and applications. *Numerical Algorithms*, 37(1-4):147–158, 2004.

[31] E.D. Dolan et J.J. Moré : Benchmarking optimization software with performance profiles. *Mathematical Programming Series A*, 91(2):201–213, 2002.

[32] E. Fitan, F. Messine et B. Nogarède : The electromagnetic actuator design problem : A general and rational approach. *IEEE Transactions on Magnetics*, 40(3): 1579–1590, 2004.

[33] J. Fontchastagner : *Résolution du problème inverse de conception d'actionneurs électromagnétiques par association de méthodes d'optimisation déterministes globales avec des modèles analytiques et numériques.* Thèse de doctorat, Institut National Polytechnique de Toulouse, 2007.

[34] J. Fontchastagner, O. Chadebec, H. Schellekens, G. Meunier et V. Mazauric : Coupling of an electrical arc model with fem for vacuum interrupter designs. *IEEE Transactions on Magnetics*, 2004.

[35] J. Fontchastagner, F. Messine et Y. Lefevre : Design of electrical rotating machines by associating deterministic global optimization algorithm with combinatorial analytical and numerical models. *IEEE Transactions on Magnetics*, 43(8):3411–3419, 2007.

[36] J. Fontchastagner, F. Messine et Y. Lefevre : A new rational way combining analytical and numerical models with a deterministic global optimization algorithm for the design of electrical rotating machines. *IEEE Transactions on Magnetics*, 43(8):3411–3419, 2007.

[37] F. Glover et M. Laguna : Tabu search. *In Handbook of Combinatorial Optimization, Vol. 3.* Kluwer Academis Publishers, Boston, 1998.

[38] L. Granvilliers et F. Benhamou : Progress in the solving of a circuit design problem. *Journal of Global Optimization*, 20(3):155–168, 2001.

[39] D. Griffiths et D. Culpin : Pi-optimal polygons. *The Mathematical Gazette*, 59(409):165–175, 1975.

[40] R. Hammer, M. Hocks, U. Kulish et D. Ratz : *Numerical Toolbox for Verified Computing I.* Springer-Verlag, Berlin, 1993.

[41] E.R. Hansen : A generalized interval arithmetic. *Lecture Notes in Computer Science*, 29:7–18, 1975.

[42] E.R. Hansen et W.G. William : *Global Optimization Using Interval Analysis.* Marcel Dekker Inc., New York, 2ème édition, 2004.

[43] P. Hansen, J.-L. Lagouanelle et F. Messine : Comparison between Baumann and admissible simplex forms in interval analysis. *Journal of Global Optimization*, 37(2):215–228, 2007.

[44] P. Hansen et N. Mladenovic : Variable neighborhood search : Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

[45] R. J. Hanson : interval arithmetic as a closed arithmetic system on a computer. Rapport technique, Jet Propulsion Lab, 1968.

[46] R. Horst et H. Tuy : *Global Optimization : Deterministic Approaches.* Springer-Verlag, Berlin, third édition, 1996.

[47] K. Ichida et Y. Fujii : An interval arithmetic method for global optimization. *Computing*, 23(1):85–97, 1979.

[48] Sun Microsystems Inc. : Fortran 95 interval arithmetic programming reference. Rapport technique, Iuniverse Inc, 2002.

[49] C. Jansson : Rigorous lower and upper bounds in linear programming. *SIAM Journal on Optimization*, 14(3):914–935, 2003.

[50] W.M. Kahan : A more complete interval arithmetic. Rapport technique, University of Michigan, 1968.

[51] R.B. Kearfott : An interval branch and bound algorithm for bound constrained optimization problems. *Journal of Global Optimization*, 2(3):259–280, 1992.

[52] R.B. Kearfott : *Rigorous Global Search : Continuous Problems*. Kluwer Academis Publishers, Dordrecht, 1996.

[53] R.B. Kearfott : Discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization. *Optimization Methods & Software*, 21(5):715–731, 2006.

[54] R.B. Kearfott et S. Hongthong : Validated linear relaxations and preprocessing : Some experiments. *SIAM Journal on Optimization*, 16(2):418–433, 2005.

[55] C. Keil : LURUPA : Rigorous error bounds in linear programming. *In Algebraic and Numerical Algorithms and Computer-assisted Proofs*, Nov 2006.

[56] C. Keil et C. Jansson : Computational experience with rigorous error bounds for the Netlib linear programming library. *Reliable Computing*, 12(6):303–321, 2006.

[57] O. Knüppel : PROFIL/BIAS - a fast interval library. *Computing*, 53(3-4):277–287, 1994.

[58] L. Kolev : Automatic computation of a linear interval enclosure. *Reliable Computing*, 7(1):17–28, 2001.

[59] R. Krawczyk et K. Nickel : The centered form in interval arithmetics - Quadratic convergence and inclusion isotonicity. *Computing*, 28(2):117–137, 1982.

[60] J.-L. Lagouanelle et F. Messine : An inclusion algorithm for finding the global optimum of a multivariate differentiable function. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 326(5):629–632, 1998.

[61] Y. Lebbah, C. Michel et M. Rueher : Efficient pruning technique based on linear relaxations. *In Proceedings of Global Optimization And Constraint Satisfaction*, volume 3478, pages 1–14, 2005.

[62] Y. Lebbah, C. Michel et M. Rueher : A rigorous global filtering algorithm for quadratic constraints. *Constraints*, 10(1):47–65, 2005.

[63] Y. Lebbah, C. Michel, M. Rueher, D. Daney et J.P. Merlet : Efficient and safe global constraints for handling numerical constraint systems. *SIAM Journal on Numerical Analysis*, 42(5):2076–2097, 2005.

[64] L. Liberti, S. Cafieri et D. Savourey : The reformulation-optimization software engine. *In Mathematical Software – ICMS 2010*, volume 6327 de *Lecture Notes in Computer Science*, pages 303–314. Springer Berlin / Heidelberg, 2010.

[65] C.D. Maranas et C.A. Floudas : Global optimization in generalized geometric programming. *Computers & Chemical Engineering*, 21(4):351–369, 1997.

[66] R. Martin, H. Shou, I. Voiculesci, A. Bowyer et G.J. Wang : Comparison of interval methods for plotting algebraic curves. *Computer Aided Geometric Design*, 19(7):553–587, 2002.

[67] G.P. McCormick : Computability of global solutions to factorable nonconvex programs. I. Convex underestimating problems. *Mathematical Programming*, 10(2):147–175, 1976.

[68] F. Messine : *Méthode d'optimisation globale basée sur l'analyse d'intervalles pour la résolution de problèmes avec contraintes*. Thèse de doctorat, Institut National Polytechnique de Toulouse, 1997.

[69] F. Messine : Extensions of affine arithmetic : Application to unconstrained global optimization. *Journal of Universal Computer Science*, 8(11):992–1015, 2002.

[70] F. Messine : Deterministic global optimization using interval constraint propagation techniques. *RAIRO-Operations Research*, 38(4):277–293, 2004.

[71] F. Messine : A deterministic global optimization algorithm for design problems. *In* C. Audet, P. Hansen et G. Savard, éditeurs : *Essays and Surveys in Global Optimization*, chapitre 10, pages 267–294. Springer, New York, 2005.

[72] F. Messine et J.-L. Lagouanelle : Enclosure methods for multivariate differentiable functions and application to global optimization. *Journal of Universal Computer Science*, 4(6):589–603, 1998.

[73] F. Messine et B. Nogarède : Optimal design of multi-airgap electrical machines : An unknown size mixed-constrained global optimization formulation. *IEEE Transactions on Magnetics*, 42(12):3847–3853, 2006.

[74] F. Messine, B. Nogarède et J.-L. Lagouanelle : Optimal design of electromechanical actuators : A new method based on global optimization. *IEEE Transactions on Magnetics*, 34(1):299–308, 1998.

[75] F. Messine et A. Touhami : A general reliable quadratic form : An extension of affine arithmetic. *Reliable Computing*, 12(3):171–192, 2006.

[76] W. Miles : *Operator overloading in C*. Thèse de doctorat, University of British Columbia, 1995.

[77] A. Mitsos, B. Chachuat et P.I. Barton : McCormick-based relaxations of algorithms. *SIAM Journal on Optimization*, 20(2):573–601, 2009.

[78] R.E. Moore : *Interval Analysis*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1966.

[79] R.E. Moore : *Methods and Applications of Interval Analysis*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1979.

[80] M.J. Mossinghoff : Enumerating isodiametric and isoperimetric polygons. *Journal of Combinatorial Theory, Series A*, 2010. à paraître.

[81] A. Neumaier : Ensemble de problèmes tests COCONUT. http ://www.mat.univie.ac.at/∼neum/glopt/coconut/Benchmark/Benchmark.html.

[82] A. Neumaier et O. Shcherbina : Safe bounds in linear and mixed-integer linear programming. *Mathematical Programming Series A*, 99(2):283–296, 2004.

[83] A. Neumaier, O. Shcherbina, W. Huyer et T. Vinko : A comparison of complete global optimization solvers. *Mathematical Programming Series B*, 103(2):335–356, 2005.

[84] J. Ninin et F. Messine : A mixed integer affine reformulation method for global optimization. *In Proceedings of TOGO - Global Optimization Workshop*, 2010.

[85] J. Ninin et F. Messine : A metaheuristic methodology based on the limitation of the memory of interval branch and bound algorithms. *Journal of Global Optimization*, en ligne et à paraître, 2010.

[86] J. Ninin, F. Messine et P. Hansen : A reliable affine relaxation method for global optimization. Rapport technique, IRIT, en soumission, 2010.

[87] J. Pál : Ein Minimumproblem für Ovale. *Mathematische Annalen*, 83:311–319, 1921.

[88] S. Perron : *Applications jointes de l'optimisation combinatoire et globale.* Thèse de doctorat, École Polytechnique de Montréal, 2004.

[89] F. Pigache, F. Messine et B. Nogarède : Optimal design of piezoelectric transformers : A rational approach based on an analytical model and a deterministic global optimization. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 54(7):1293–1302, 2007.

[90] H. Ratschek et J. Rokne : *Computer Methods for the Range of Functions.* Ellis Horwood Ltd, Chichester, 1984.

[91] H. Ratschek et J. Rokne : *New Computer Methods for Global Optimization.* Ellis Horwood Ltd, Chichester, 1988.

[92] F. Reuleaux : *The Kinematics of Machinery.* Dover, New York, 1963.

[93] N. Revol, K. Makino et A. Berz : Taylor models and floating-point arithmetic : proof that arithmetic operations are validated in COSY. *Journal of Logic And Algebraic Programming*, 64(1):135–154, 2005.

[94] S.M. Rump : INTLAB - INTerval LABoratory. *In Developments in Reliable Computing*, pages 77–104. Kluwer Academis Publishers, Dordrecht, 1999.

[95] O. Shcherbina, A. Neumaier, D. Sam-Haroud, X.H. Vu et T.V. Nguyen : Benchmarking global optimization and constraint satisfaction codes. *Lecture Notes in Computer Science*, 2861:211–222, 2003.

[96] H.D. Sherali et W.P. Adams : *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems.* Kluwer Academis Publishers, Dordrecht, 1999.

[97] H.D. Sherali et J. Desai : On solving polynomial, factorable, and black-box optimization problems using the RLT methodology. *In* Charles Audet, Pierre Hansen et Gilles Savard, éditeurs : *Essays and Surveys in Global Optimization*, pages 131–163. Springer US, 2005.

[98] H.D. Sherali et L. Liberti : Reformulation-linearization technique for global optimization. *In Encyclopedia of Optimization*, volume 18, pages 3263–3268. Springer-Verlag, 2009.

[99] H.D. Sherali et H.J. Wang : Global optimization of nonconvex factorable programming problems. *Mathematical Programming*, 89(3):459–478, Jan 2001.

[100] S. Skelboe : Computation of rational interval functions. *BIT Numerical Mathematics*, 14(1):87–95, 1974.

[101] J. Stolfi et L. de Figueiredo : Self-validated numerical methods and applications. *Monograph for 21st Brazilian Mathematics Colloquium*, 1997.

[102] M. Tawarmalani et N.V. Sahinidis : Global optimization of mixed-integer nonlinear programs : A theoretical and computational study. *Mathematical Programming Series A*, 99(3):563–591, 2004.

[103] A. Touhami : Utilisation et extension de l'arithmétique affine dans les algorithmes déterministes d'optimisation globale. Mémoire de D.E.A., Institut National Polytechnique de Toulouse, 2002.

[104] P. Van Hentenryck, L. Michel et Y. Deville : *Numerica : a Modelling Language for Global Optimization*. MIT Press, 1997.

[105] S. Vincze : On a geometrical extremum problem. *Acta Universitatis Szegediensis*, 12:136–142, 1950.

[106] T. Vinko, J.-L. Lagouanelle et T. Csendes : A new inclusion function for optimization : Kite - the one dimensional case. *Journal of Global Optimization*, 30(4):435–456, 2004.

[107] X.-H. Vu, D. Sam-Haroud et B. Faltings : Enhancing numerical constraint propagation using multiple inclusion representations. *Annals of Mathematics and Artificial Intelligence*, 55(3-4):295–354, 2009.

[108] I.M. Yaglom et V. G. Boltyanskii : *Convex figures*. Holt, Rinehart and Winston. Translated from Russian by P.J. Kelly and L.F. Walton, New York, 1961.

Annexe A

Détails des résultats numériques de la section 3.5

Les tableaux suivants compilent la totalité et les détails des résultats numériques obtenus pour créer les tableaux 3.3, 3.4, 3.5 et 3.6 du chapitre 3. Les conditions dans lesquelles ils ont été effectués sont décrites en introduction de la section 3.5

IBBA (algorithme 2) a été testé sur 74 problèmes issus de la librairie 1 du site web COCONUT [81], en incluant ou non différentes techniques d'accélération, afin de déterminer l'impact de celle-ci sur la vitesse de convergence et le comportement de l'algorithme.

Le nom des problèmes est inscrit dans la première colonne des tableaux. Les colonnes N et M représentent le nombre de variables et le nombre de contraintes pour chaque problème. La colonne 'ok ?' indique un 'T' lorsque l'algorithme a résolu le problème et un 'F' lorsque l'algorithme n'a pas terminé en moins d'une heure ou lorsqu'il a atteint la limite imposée sur le nombre d'éléments pouvant être stockés dans $\mathcal{L}$, fixé à 2 000 000 éléments. La colonne 'Minorant' indique le plus grand minorant certifié sur la valeur du minimum global et la colonne 'Majorant' indique la meilleure solution connue, c'est-à-dire un majorant de la valeur du minimum global. 'Iter' indique le nombre total d'itérations de la boucle principale de *IBBA* (algorithme 2) et 't (s)' indique le temps de calcul en seconde. La dernière colonne 'max $|\mathcal{L}|$' désigne la taille maximale de la structure de données $\mathcal{L}$.

Les algorithmes utilisés pour les tests sont les suivants :

1. ***IBBA+rART***$_{rAF2}$***+CP*** **:** *IBBA* (algorithme 2) incluant la technique de propagation des contraintes (algorithme 3) et la technique de reformulation affine robuste (algorithme 9) utilisant l'arithmétique affine robuste rAF2 (section 2.5.2).
2. ***IBBA+rART***$_{rAF2}$ **:** *IBBA* (algorithme 2) incluant seulement la technique de reformulation affine robuste (algorithme 9) utilisant l'arithmétique affine rAF2 (section 2.5.2).
3. ***IBBA+CP*** **:** *IBBA* (algorithme 2) incluant la technique de propagation des contraintes (algorithme 3).

4. ***IBBA+ART***$_{AF2\_primal}$***+CP*** **:** *IBBA* (algorithme 2) incluant la technique de propagation des contraintes (algorithme 3) et la technique de reformulation affine (algorithme 8) utilisant l'arithmétique affine AF2 avec une résolution du programme linéaire par l'algorithme primal.
5. ***IBBA+ART***$_{AF2\_dual}$***+CP*** **:** *IBBA* (algorithme 2) incluant la technique de propagation des contraintes (algorithme 3) et la technique de reformulation affine (algorithme 8) utilisant l'arithmétique affine AF2 avec une résolution du programme linéaire par l'algorithme dual.
6. ***IBBA+rART***$_{AF2}$***+CP*** **:** *IBBA* (algorithme 2) incluant la technique de propagation des contraintes (algorithme 3) et la technique de reformulation affine robuste (algorithme 9) utilisant l'arithmétique affine AF2.
7. ***IBBA+rART***$_{fAF2}$***+CP*** **:** *IBBA* (algorithme 2) incluant la technique de propagation des contraintes (algorithme 3) et la technique de reformulation affine robuste (algorithme 9) utilisant l'arithmétique affine robuste fAF2 (section 2.5.3).
8. ***IBBA+rART***$_{sAF2}$***+CP*** **:** *IBBA* (algorithme 2) incluant la technique de propagation des contraintes (algorithme 3) et la technique de reformulation affine robuste (algorithme 9) utilisant l'arithmétique affine robuste sAF2 (section 2.5.1).

A.1 Résultats de $IBBA+rART_{rAF2}+CP$

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
hs071	4	2	T	17.0140171933527	17.014017363492872	804	1.04	735
ex2_1_1	5	1	T	-17.00000003961484	-16.99999986961484	151	0.23	20
ex2_1_2	6	2	T	-213.0000018338394	-212.99999970383942	105	0.18	105
ex2_1_3	13	9	T	-15.000000014376148	-14.99999986437615	266	0.52	260
ex2_1_4	6	5	T	-11.000000043503567	-10.999999933503567	250	0.27	223
ex2_1_5	10	11	T	-268.01463416730075	-268.0146314871544	263	0.66	203
ex2_1_6	10	5	T	-39.00000004727329	-38.99999965727329	285	0.69	40
ex2_1_7	20	10	T	-4150.4101750826184	-4150.410133578517	1574	16.75	1142
ex2_1_8	24	10	T	15638.999865821344	15639.000022211345	1916	26.77	1916
ex2_1_9	10	1	T	-0.3750000101909464	-0.3750000001909464	60007	154.02	25762
ex2_1_10	20	10	T	49318.01747045439	49318.01796363457	636	5.91	636
ex3_1_1	8	6	T	7049.247950045217	7049.248020537698	131195	115.92	17898
ex3_1_2	5	6	T	-30665.53897927148	-30665.538672616094	111	0.19	44
ex3_1_3	6	6	T	-310.00000129459334	-309.9999981945934	182	0.24	133
ex3_1_4	3	3	T	-4.000000025098839	-3.999999985098839	187	0.25	23
ex4_1_8	2	1	T	-16.73889332479729	-16.738893157408355	128	0.11	61
ex4_1_9	2	2	T	-5.5080133221501874	-5.508013267070055	157	0.17	15
ex5_2_2_case1	9	6	T	-400.0000037442501	-399.9999997442501	5233	8.05	2634
ex5_2_2_case2	9	6	T	-600.0000058158159	-599.999999815816	9180	14.73	3681
ex5_2_2_case3	9	6	T	-750.0000074519787	-749.9999999519787	2255	3.44	1534
ex5_2_4	7	6	T	-450.00000438201044	-449.99999988201045	9848	11.30	9076
ex5_4_2	8	6	T	7512.23006938046	7512.2301445027615	201630	121.45	9324

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex6_1_1	9	6	F	-0.020615993431239695	$+\infty$	1500000	3775.80	379077
ex6_1_2	5	3	T	-0.032463794582323124	-0.03246378458232312	108	0.26	46
ex6_1_3	13	9	F	-0.42677049501066904	$+\infty$	1000000	3913.87	365054
ex6_1_4	7	4	T	-0.29454129783452365	-0.29454128783452365	1622	2.70	692
ex6_2_5	10	3	F	-7036.008545972475	$+\infty$	800000	4055.01	800000
ex6_2_6	4	1	T	-2.612524607930867E-6	-2.602524607930867E-6	922664	1575.43	81602
ex6_2_7	10	3	F	-66.22734636738744	$+\infty$	500000	4036.90	500000
ex6_2_8	4	1	T	-0.027006358842591345	-0.027006348842591343	265276	457.87	18364
ex6_2_9	5	2	T	-0.03406619411329543	-0.03406618411329543	203775	522.57	15583
ex6_2_10	7	3	F	-3.4877755579697184	-3.0519497528589663	1200000	3775.14	697120
ex6_2_11	4	1	T	-2.682392735172946E-6	-2.672392735172946E-6	83487	140.51	6996
ex6_2_12	5	2	T	0.2891947384735326	0.2891947484735326	58231	112.58	4096
ex6_2_13	7	3	F	-0.27559514312663346	-0.21620660088298077	1500000	3650.76	797908
ex6_2_14	5	2	T	-0.6953579385812835	-0.6953579285812834	95170	207.78	16715
ex7_2_1	7	14	T	1227.2260665512837	1227.2260788235446	8419	24.72	5288
ex7_2_2	6	5	T	-0.38881144934638917	-0.3888114393463892	531	0.87	469
ex7_2_3	8	6	F	7049.2479614466765	7049.277305603027	2200000	3716.02	107484
ex7_2_5	5	6	T	10122.493217568921	10122.493318793855	186	0.40	85
ex7_2_6	3	1	T	-83.24972967399799	-83.2497288415007	1319	1.23	319
ex7_2_10	11	9	T	0.09999999595450439	0.10000000595450438	1417	2.19	1262
ex7_3_1	4	7	T	0.34173955182411774	0.34173956182411774	1536	3.50	705
ex7_3_2	4	7	T	1.0898639601992741	1.0898639710979138	141	0.28	55
ex7_3_3	5	8	T	0.8175290406218059	0.8175290506218059	373	0.66	206
ex7_3_4	12	17	F	0.0E+0	$+\infty$	1000000	3971.40	585427
ex7_3_5	13	15	F	-1.430511474609375E-6	$+\infty$	500000	4259.44	458420
ex7_3_6	17	17	T	9.9999999E+299	$+\infty$	1	0.14	0

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex8_1_7	5	5	T	0.02931082245928056	0.02931083245928056	1432	2.64	1153
ex8_1_8	6	5	T	-0.38881144934638917	-0.3888114393463892	531	0.87	469
ex9_2_1	11	9	T	16.99999983	17.0	64	0.26	3
ex9_2_2	11	11	F	99.99987284350027	$+\infty$	4700000	3602.48	1568227
ex9_2_3	17	15	T	-1.0E-8	0.0E+0	156	0.50	11
ex9_2_4	9	7	T	0.4999999904656613	0.5000000004656613	49	0.25	5
ex9_2_5	9	7	T	4.999999976410212	5.000000026410213	136	0.44	56
ex9_2_6	17	12	F	-1.5098876953125002	56.3203125	1200000	3756.22	963638
ex9_2_7	11	9	T	16.99999983	17.0	64	0.35	3
ex14_1_1	3	4	T	-1.907867963007207E-8	-9.07867963007207E-9	301	0.54	15
ex14_1_2	6	9	T	-7.559065880604128E-9	2.4409341193958722E-9	24166	54.58	7294
ex14_1_3	3	4	T	-2.049408845572937E-9	7.950591154427063E-9	91	0.26	20
ex14_1_5	6	6	T	-3.075417549157069E-9	6.924582450842931E-9	1752	2.99	340
ex14_1_6	9	15	T	-2.4641053797033124E-9	7.535894620296688E-9	2531	12.45	380
ex14_1_7	10	17	F	-10.664982241554853	3234.9943010627066	1100000	3703.00	8556
ex14_1_8	3	4	T	-1.2065084208750782E-9	8.793491579124922E-9	77	0.25	23
ex14_1_9	2	2	T	-1.4308548406939394E-8	-4.308548406939394E-9	223	0.35	36
ex14_2_1	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	16786	36.73	2559
ex14_2_2	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	1009	1.39	160
ex14_2_3	6	9	T	-5.6631913100579824E-9	4.336808689942018E-9	47673	173.28	9706
ex14_2_4	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	30002	127.56	8758
ex14_2_5	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	2041	3.70	188
ex14_2_6	5	7	T	-5.6631913100579824E-9	4.336808689942018E-9	74630	237.56	24002
ex14_2_7	6	9	F	0.0E+0	0.21827872842550277	700000	3841.44	256647
ex14_2_8	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	10044	19.13	358
ex14_2_9	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	6582	14.59	603
Moyenne lorsque '**T**'		61				37556.7	69.3	4657.21

A.2 Résultats de $IBBA+rART_{rAF2}$

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\lvert\mathcal{L}\rvert$
hs071	4	2	T	17.014017242775985	17.01401741291616	1580	2.44	1095
ex2_1_1	5	1	T	-17.00000003961484	-16.99999986961484	151	0.32	20
ex2_1_2	6	2	T	-213.0000021085796	-212.99999997857958	289	0.46	179
ex2_1_3	13	9	T	-15.000000028928063	-14.999999878928065	352	0.74	272
ex2_1_4	6	5	T	-11.000000052258	-10.999999942258	641	0.74	293
ex2_1_5	10	11	T	-268.0146340533155	-268.01463137316915	844	1.98	504
ex2_1_6	10	5	T	-39.00000004727329	-38.99999965727329	286	0.77	40
ex2_1_7	20	10	T	-4150.410174739903	-4150.410133235801	1569	16.25	1143
ex2_1_8	24	10	T	15638.99986158569	15639.00001797569	3908	53.37	3804
ex2_1_9	10	1	T	-0.37500001431009893	-0.37500000431009894	66180	160.10	26183
ex2_1_10	20	10	T	49318.01746572001	49318.01795890019	938	8.81	421
ex3_1_1	8	6	T	7049.247949443177	7049.248019935658	81818	137.02	25201
ex3_1_2	5	6	T	-30665.53897144459	-30665.5386647892	144	0.36	106
ex3_1_3	6	6	T	-310.00000251512943	-309.9999994151294	243	0.55	173
ex3_1_4	3	3	T	-4.000000008335032	-3.9999999683350324	171	0.37	20
ex4_1_8	2	1	T	-16.73889338878477	-16.738893221395837	137	0.32	57
ex4_1_9	2	2	T	-5.5080133221501874	-5.508013267070055	171	0.19	18
ex5_2_2_case1	9	6	F	-618.75	$+\infty$	2300000	3699.67	132773
ex5_2_2_case2	9	6	F	-1675.0	$+\infty$	2200000	3646.14	157792
ex5_2_2_case3	9	6	F	-825.988051470589	$+\infty$	2300000	3682.61	153000
ex5_2_4	7	6	T	-450.00000933851185	-450.0000048385118	128303	142.42	93644
ex5_4_2	8	6	T	7512.23006938622	7512.230144508521	8714	12.72	2536

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex6_1_1	9	6	F	-32.23589702915751	$+\infty$	1600000	3756.88	802520
ex6_1_2	5	3	T	-0.0324637886094911	-0.032463778609491095	1813	2.39	338
ex6_1_3	13	9	F	-32.191620233919415	$+\infty$	900000	3704.39	432872
ex6_1_4	7	4	T	-0.2945413105437519	-0.2945413005437519	148480	262.65	36838
ex6_2_5	10	3	F	-9123.815284758357	$+\infty$	800000	3934.43	620850
ex6_2_6	4	1	F	-2.6129448684113553E-6	-2.602065528212183E-6	2100000	3719.93	122552
ex6_2_7	10	3	F	-97.96423196449967	$+\infty$	500000	3973.21	379034
ex6_2_8	4	1	T	-0.027006358842546492	-0.02700634884254649	634377	1122.06	30074
ex6_2_9	5	2	F	-0.03420672913753995	-0.034066153463143233	1500000	3700.92	106667
ex6_2_10	7	3	F	-6.625714752975224	-3.0510283773510256	1300000	3872.20	366432
ex6_2_11	4	1	T	-2.682392733799045E-6	-2.672392733799045E-6	214420	346.71	13651
ex6_2_12	5	2	T	0.28919473849472554	0.28919474849472553	1096081	2136.20	35301
ex6_2_13	7	3	F	-0.942992601222289	-0.2161410232408119	1600000	3605.98	542825
ex6_2_14	5	2	T	-0.695357941593758	-0.6953579315937579	450059	956.88	42781
ex7_2_1	7	14	T	1227.2260572459415	1227.2260695182021	18037	50.64	7699
ex7_2_2	6	5	T	-0.38881144927574157	-0.3888114392757416	4312	5.64	1011
ex7_2_3	8	6	F	2275.78125	$+\infty$	2300000	3684.73	223478
ex7_2_5	5	6	T	10122.493237968429	10122.493339193363	249	0.52	108
ex7_2_6	3	1	T	-83.24972971138992	-83.24972887889263	2100	1.89	324
ex7_2_10	11	9	T	0.09999999000000001	0.1	2605	3.96	1577
ex7_3_1	4	7	T	0.34173954305123394	0.34173955305123393	2713	6.21	701
ex7_3_2	4	7	T	1.0898639347205232	1.0898639456191627	2831	3.05	305
ex7_3_3	5	8	T	0.8175290383697285	0.8175290483697285	1104	1.74	388
ex7_3_4	12	17	F	0.0E+0	$+\infty$	800000	3756.89	434
ex7_3_5	13	15	F	-3.5762786865234375E-6	$+\infty$	500000	4291.36	280328
ex7_3_6	17	17	T	9.9999999E+299	$+\infty$	84	5.55	6

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex8_1_7	5	5	T	0.0293108189786817	0.029310828978681702	6183	12.25	2096
ex8_1_8	6	5	T	-0.38881144927574157	-0.3888114392757416	4312	5.65	1011
ex9_2_1	11	9	F	16.999999796058723	$+\infty$	1800000	3637.95	129674
ex9_2_2	11	11	F	99.99847408616915	$+\infty$	2000000	3653.05	433060
ex9_2_3	17	15	F	-3.9968031480544036E-14	$+\infty$	1500000	3740.15	66689
ex9_2_4	9	7	T	0.4999999919790607	0.5000000019790607	4682	7.96	495
ex9_2_5	9	7	T	4.999999946274722	4.999999996274722	6331	12.69	542
ex9_2_6	17	12	F	-1.5707811054430518	$+\infty$	1000000	3611.39	684409
ex9_2_7	11	9	F	16.999999783335034	$+\infty$	1700000	3643.63	120727
ex14_1_1	3	4	T	-1.0211770832609177E-8	-2.1177083260917788E-10	1728	2.75	130
ex14_1_2	6	9	T	-1.187036901410071E-8	-1.870369014100709E-9	59677	206.88	11797
ex14_1_3	3	4	F	2.2423781589568552E-9	228880.84549573255	8000000	3629.02	65234
ex14_1_5	6	6	T	-3.075417549157069E-9	6.924582450842931E-9	3961	6.38	617
ex14_1_6	9	15	T	-6.69052368067043E-9	3.3094763193295702E-9	6326	26.61	2411
ex14_1_7	10	17	F	-120.09538849356366	3234.9943010627066	600000	4155.17	6935
ex14_1_8	3	4	T	-9.87445059880822E-9	1.2554940119177991E-10	2011	2.30	48
ex14_1_9	2	2	T	-5.218028628659333E-9	4.781971371340667E-9	23465	17.84	31
ex14_2_1	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	30436	64.13	3305
ex14_2_2	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	2671	3.64	306
ex14_2_3	6	9	T	-1.3263826201159647E-9	8.673617379884035E-9	70967	252.31	12143
ex14_2_4	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	62245	274.42	2130
ex14_2_5	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	5821	11.75	401
ex14_2_6	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	138654	407.27	28716
ex14_2_7	6	9	F	0.0E+0	0.29103830456733703	800000	4021.63	270824
ex14_2_8	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	31840	57.17	2149
ex14_2_9	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	19474	40.44	1967
Moyenne lorsque '**T**'		52				64547.85	131.89	7636.65

A.3 Résultats de *IBBA+CP*

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
hs071	4	2	T	17.014017120851217	17.01401729099139	9558537	722.34	1385701
ex2_1_1	5	1	T	-17.00000008385266	-16.999999913852661	26208	1.17	8826
ex2_1_2	6	2	T	-213.0000018338394	-212.99999970383942	105	0.00	105
ex2_1_3	13	9	F	-16.49462890625	-14.15625	2004691	106.02	1999999
ex2_1_4	6	5	T	-11.000000041827185	-10.999999931827187	5123	0.25	669
ex2_1_5	10	11	T	-268.0146342105952	-268.01463153044886	172195	32.70	121739
ex2_1_6	10	5	T	-39.00000028196658	-38.99999989196658	5109625	565.22	1052710
ex2_1_7	20	10	F	-4718.125	1.0E+300	7075425	1735.14	1999999
ex2_1_8	24	10	F	5394.75	1.0E+300	2005897	280.95	1999999
ex2_1_9	10	1	F	-0.5390625	-0.2705078125	1999999	93.57	1999999
ex2_1_10	20	10	F	47059.499497639713	$+\infty$	1999999	635.95	1999999
ex3_1_1	8	6	F	7040.771484375	$+\infty$	38000000	3604.46	1594391
ex3_1_2	5	6	T	-30665.53885394586	-30665.538547290474	6571	0.44	636
ex3_1_3	6	6	T	-310.00000072089864	-309.9999976208987	4321	0.21	365
ex3_1_4	3	3	T	-4.000000023236193	-3.9999999832361936	21096	1.06	857
ex4_1_8	2	1	T	-16.73889331724288	-16.738893149853947	78417	2.31	33227
ex4_1_9	2	2	T	-5.5080133221501874	-5.508013267070055	49678	6.38	239
ex5_2_2_case1	9	6	F	-424.21875	-90.625	4266494	308.90	1999999
ex5_2_2_case2	9	6	F	-600.0000486480653	$+\infty$	7027892	529.41	1999999
ex5_2_2_case3	9	6	F	-780.46875	-698.4375	3671986	257.71	1999999
ex5_2_4	7	6	F	-2192.727048976594	146.8749725588856	3338206	510.99	1999999
ex5_4_2	8	6	F	7512.21458914264	$+\infty$	43800000	3606.12	1895436

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex6_1_1	9	6	F	-0.022761432589189122	$+\infty$	5270186	2805.03	1999999
ex6_1_2	5	3	T	-0.032463794582323124	-0.03246378458232312	15429	0.83	5002
ex6_1_3	13	9	F	-0.4387313777963412	$+\infty$	4534626	3233.97	1999999
ex6_1_4	7	4	F	-0.2945417707717488	-0.29454128783452365	2444266	204.92	1999999
ex6_2_5	10	3	F	-5831.2075381299055	$+\infty$	1999999	192.80	1999999
ex6_2_6	4	1	F	-0.18853329619925407	7.518656701677173E-7	2097277	124.56	1999999
ex6_2_7	10	3	F	-60.251669786928054	$+\infty$	1999999	229.94	1999999
ex6_2_8	4	1	F	-0.25320217897795727	-0.027005901014119704	2003020	118.81	1999999
ex6_2_9	5	2	F	-0.04354612167732763	-0.03406616995572209	3724203	369.78	1999999
ex6_2_10	7	3	F	-4.053226989492883	-3.0519497528589663	1999999	241.17	1999999
ex6_2_11	4	1	F	-0.029042724521662233	-2.354018679540104E-6	2729823	149.66	1999999
ex6_2_12	5	2	F	0.28817364584160065	0.28919475056726073	2975037	202.77	1999999
ex6_2_13	7	3	F	-0.4312244616345242	-0.2162084236780082	2007671	332.47	1999999
ex6_2_14	5	2	T	-0.6953579444356377	-0.6953579344356376	8446077	988.14	1954789
ex7_2_1	7	14	F	1190.794234242079	1386.221788358218	9324644	2512.97	1999999
ex7_2_2	6	5	F	-0.38881152527729373	-0.3888114393776936	4990110	1031.30	1999999
ex7_2_3	8	6	F	6874.658203125	$+\infty$	41000000	3607.35	856049
ex7_2_5	5	6	T	10122.493124846364	10122.493226071296	6000	0.67	552
ex7_2_6	3	1	F	-83.24973320357533	-83.24972927795261	7022520	326.38	1999999
ex7_2_10	11	9	T	0.09999999595450439	0.10000000595450438	1417	0.09	1262
ex7_3_1	4	7	T	0.3417395510814393	0.34173956108143926	33347	4.23	2454
ex7_3_2	4	7	T	1.0898639601992741	1.0898639710979138	141	0.08	55
ex7_3_3	5	8	T	0.8175290381823509	0.817529048182351	18603	2.14	2814
ex7_3_4	12	17	F	-5.0	$+\infty$	3194446	467.81	1999999
ex7_3_5	13	15	F	-300132.92040438	$+\infty$	3017872	513.88	1999999
ex7_3_6	17	17	T	9.9999999E+299	$+\infty$	1	0.00	0

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex8_1_7	5	5	F	0.02930845923000994	0.029310827803941414	3807889	395.30	1999999
ex8_1_8	6	5	F	-0.38881152527729373	-0.3888114393776936	4990110	1029.00	1999999
ex9_2_1	11	9	T	16.99999983	17.0	161	0.02	8
ex9_2_2	11	11	F	99.9999046325911	$+\infty$	5902793	314.66	1999999
ex9_2_3	17	15	T	-8.69030631316491E-9	1.3096936868350894E-9	884	0.15	253
ex9_2_4	9	7	T	0.4999999904656613	0.5000000004656613	77	0.00	4
ex9_2_5	9	7	T	4.9999999490708084	4.999999999070808	51303	7.59	12088
ex9_2_6	17	12	F	-4.03857421875	8.09033203125	2895007	233.85	1999999
ex9_2_7	11	9	T	16.99999983	17.0	161	0.02	8
ex14_1_1	3	4	T	-1.9078677963007207E-8	-9.078677963007207E-9	367	0.13	17
ex14_1_2	6	9	T	-8.829666140592808E-9	1.1703338594071921E-9	619905	145.68	152338
ex14_1_3	3	4	T	-2.049408845572937E-9	7.950591154427063E-9	94	0.00	22
ex14_1_5	6	6	T	-1.6085843659315443E-8	-6.085843659315443E-9	165381	18.06	17598
ex14_1_6	9	15	T	-7.869563819470148E-9	2.1304361805298524E-9	42139	8.88	3456
ex14_1_7	10	17	F	-3.097986064554242	3234.9943010627066	9600000	3635.90	1074699
ex14_1_8	3	4	T	-5.930893681546904E-9	4.0691063184530965E-9	98	0.01	26
ex14_1_9	2	2	T	-1.3892569759723548E-8	-3.892569759723548E-9	1300	0.05	121
ex14_2_1	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	12017408	1683.62	336389
ex14_2_2	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	8853	0.67	961
ex14_2_3	6	9	F	0.0E+0	1.0658141036401503E-4	13800000	3622.13	717365
ex14_2_4	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	1975320	455.49	152220
ex14_2_5	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	18821	1.92	398
ex14_2_6	5	7	F	0.0E+0	9.094947017729282E-3	9543033	2124.91	1999999
ex14_2_7	6	9	F	0.0E+0	0.14551915228366851	7678896	2844.60	1999999
ex14_2_8	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	2085323	279.49	124944
ex14_2_9	4	5	T	-3.4947869650869736E-9	6.505213034913027E-9	463414	70.83	33552
Moyenne lorsque '**T**'		37				1108213.51	135.16	146119.05

A.4 Résultats de $IBBA+ART_{AF2\_primal}+CP$

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
hs071	4	2	T	17.01401711563896	17.014017285779132	700	1.03	105
ex2_1_1	5	1	T	-17.00000017	-17.0	8	0.25	2
ex2_1_2	6	2	T	-213.00000213	-213.0	1	0.21	0
ex2_1_3	13	9	T	-15.00000015	-15.0	3	0.18	2
ex2_1_4	6	5	T	-11.00000011	-11.0	1	0.16	0
ex2_1_5	10	11	T	-268.01463422162015	-268.0146315414738	73	0.24	1
ex2_1_6	10	5	T	-39.00000039	-39.0	9	0.19	6
ex2_1_7	20	10	T	-4150.410175432363	-4150.4101339282615	792	1.44	28
ex2_1_8	24	10	T	15638.99984361	15639.0	133	0.23	6
ex2_1_9	10	1	T	-0.37500001	-0.375	37810	39.74	3117
ex2_1_10	20	10	T	49318.017465231416	49318.0179584116	277	0.56	3
ex3_1_1	8	6	T	7049.24801244333	7049.24808293581	35533	41.32	16460
ex3_1_2	5	6	T	-30665.53898580842	-30665.538679153032	59	0.21	17
ex3_1_3	6	6	T	-310.0000031	-310.0	106	0.27	11
ex3_1_4	3	3	T	-4.00000004	-4.0	92	0.27	9
ex4_1_8	2	1	T	-16.738893334808324	-16.73889316741939	57	0.16	22
ex4_1_9	2	2	T	-5.508013332453526	-5.508013277373394	139	0.25	15
ex5_2_2_case1	9	6	T	-400.0000040709401	-400.00000007094013	4941	6.70	2417
ex5_2_2_case2	9	6	T	-600.0000060375432	-600.0000000375433	8983	11.99	3502
ex5_2_2_case3	9	6	T	-750.0000075432918	-750.0000000432917	1846	2.49	1163
ex5_2_4	7	6	T	-450.0000045305422	-450.0000000305422	9440	8.08	8813
ex5_4_2	8	6	T	7512.230069378202	7512.230144500503	3680	3.84	1615

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex6_1_1	9	6	F	-0.02144205037103049	-0.017579414056123347	2000000	3794.47	491494
ex6_1_2	5	3	T	-0.032463794582323124	-0.03246378458232312	138	0.30	62
ex6_1_3	13	9	F	-0.451381663757428	-0.3243488170584509	1400000	3871.08	471098
ex6_1_4	7	4	T	-0.29454129783452365	-0.29454128783452365	2726	3.43	1160
ex6_2_5	10	3	F	-5831.2075381299055	-70.60034485943237	1999999	2005.80	1999999
ex6_2_6	4	1	T	-2.612527082284922E-6	-2.6025270822849222E-6	1538181	1285.85	130472
ex6_2_7	10	3	F	-60.97612838433849	-0.14532645307029845	1800000	3637.42	1800000
ex6_2_8	4	1	T	-0.02700635884306208	-0.027006348843062077	443676	420.40	29620
ex6_2_9	5	2	T	-0.03406619411329543	-0.03406618411329543	334939	342.10	24751
ex6_2_10	7	3	F	-3.1600327229806137	-3.0519642956859942	4333634	2931.14	1999999
ex6_2_11	4	1	T	-2.682395168712427E-6	-2.672395168712427E-6	135634	124.80	12380
ex6_2_12	5	2	T	0.2891947384735326	0.2891947484735326	92057	76.53	6785
ex6_2_13	7	3	F	-0.23526029049254815	-0.21620943687698135	4170289	2362.08	1999999
ex6_2_14	5	2	T	-0.6953579446244857	-0.6953579346244857	99275	90.03	8182
ex7_2_1	7	14	T	1227.2259651736892	1227.225977445949	10756	18.11	6478
ex7_2_2	6	5	T	-0.3888114445298021	-0.3888114345298021	583	0.76	450
ex7_2_3	8	6	F	6895.3125	$+\infty$	4700000	3684.41	140997
ex7_2_5	5	6	T	10122.493113234434	10122.493214459366	322	0.47	107
ex7_2_6	3	1	T	-83.24972976977953	-83.24972893728224	2083	1.75	340
ex7_2_10	11	9	T	0.09999999000000001	0.1	37	0.22	37
ex7_3_1	4	7	T	0.3417395431237205	0.3417395531237205	1506	2.28	168
ex7_3_2	4	7	T	1.0898639586247951	1.089863969523435	82	0.24	30
ex7_3_3	5	8	T	0.8175290381865713	0.8175290481865713	204	0.42	132
ex7_3_4	12	17	F	0.0E+0	$+\infty$	3414184	3465.06	1999999
ex7_3_5	13	15	F	0.0E+0	$+\infty$	3439246	3479.46	1999999
ex7_3_6	17	17	T	9.9999999E+299	$+\infty$	1	0.16	0

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex8_1_7	5	5	T	0.02931082178569189	0.029310831785691892	1115	1.46	446
ex8_1_8	6	5	T	-0.3888114445298021	-0.3888114345298021	583	0.82	450
ex9_2_1	11	9	T	16.99999983	17.0	64	0.23	3
ex9_2_2	11	11	T	99.99969382468747	99.99969482468441	589806	329.03	98305
ex9_2_3	17	15	T	-1.0E-8	0.0E+0	35	0.22	5
ex9_2_4	9	7	T	0.4999999904656613	0.5000000004656613	49	0.30	4
ex9_2_5	9	7	T	4.999999957789511	5.00000000778951	133	0.33	30
ex9_2_6	17	12	F	-1.341126559073465	-1.0	2800000	3733.08	1239563
ex9_2_7	11	9	T	16.99999983	17.0	64	0.24	3
ex14_1_1	3	4	T	2.4999994110000057E+9	2.499999436E+9	2	0.17	1
ex14_1_2	6	9	T	-1.4595769226953723E-8	-4.595769226953723E-9	23769	20.25	7144
ex14_1_3	3	4	T	-3.5884438251321173E-9	6.411556174867883E-9	73	0.22	14
ex14_1_5	6	6	T	-1.2864452094072131E-8	-2.864452094072131E-9	629	1.06	52
ex14_1_6	9	15	T	2.4999999745808143E+9	2.4999999995808143E+9	2	0.31	1
ex14_1_7	10	17	F	-3.680062693554289	2042.9013648706106	3900000	3660.68	341793
ex14_1_8	3	4	T	-1.3559221983878366E-8	-3.559221983878366E-9	73	0.27	26
ex14_1_9	2	2	T	-1.8273710585355795E-8	-8.273710585355795E-9	220	0.35	48
ex14_2_1	5	7	T	-1.0E-8	0.0E+0	33377	40.10	2027
ex14_2_2	4	5	T	-1.0E-8	0.0E+0	1292	1.14	87
ex14_2_3	6	9	T	-1.0E-8	0.0E+0	112983	176.29	10047
ex14_2_4	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	195226	190.41	10170
ex14_2_5	4	5	T	-1.0E-8	0.0E+0	2563	2.31	114
ex14_2_6	5	7	T	-1.0E-8	0.0E+0	171158	238.47	45538
ex14_2_7	6	9	F	0.0E+0	0.10913936421275139	1900000	3792.93	632962
ex14_2_8	4	5	T	-1.0E-8	0.0E+0	18853	19.03	461
ex14_2_9	4	5	T	-1.0E-8	0.0E+0	13194	14.22	1496
Moyenne lorsque '**T**'			62			63421.71	56.85	7015.16

A.5 Résultats de $IBBA+ART_{AF2\_dual}+CP$

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
hs071	4	2	T	17.01401711563896	17.014017285779132	701	1.06	104
ex2_1_1	5	1	T	-17.00000017	-17.0	8	0.19	2
ex2_1_2	6	2	T	-213.00000213	-213.0	1	0.20	0
ex2_1_3	13	9	T	-15.00000015	-15.0	3	0.28	2
ex2_1_4	6	5	T	-11.00000011	-11.0	1	0.15	0
ex2_1_5	10	11	T	-268.01463422162015	-268.0146315414738	73	0.23	1
ex2_1_6	10	5	T	-39.00000039	-39.0	9	0.19	6
ex2_1_7	20	10	T	-4150.410175432363	-4150.4101339282615	792	1.45	28
ex2_1_8	24	10	T	15638.99984361	15639.0	133	0.22	6
ex2_1_9	10	1	T	-0.37500001	-0.375	37791	36.06	3118
ex2_1_10	20	10	T	49318.017465231416	49318.0179584116	277	0.57	3
ex3_1_1	8	6	T	7049.24801244333	7049.24808293581	35470	36.63	16413
ex3_1_2	5	6	T	-30665.53898580842	-30665.538679153032	59	0.25	17
ex3_1_3	6	6	T	-310.0000031	-310.0	106	0.28	11
ex3_1_4	3	3	T	-4.00000004	-4.0	92	0.28	9
ex4_1_8	2	1	T	-16.738893334808324	-16.73889316741939	57	0.17	22
ex4_1_9	2	2	T	-5.508013332453526	-5.508013277373394	139	0.31	15
ex5_2_2_case1	9	6	T	-400.0000040709401	-400.00000007094013	4941	5.73	2417
ex5_2_2_case2	9	6	T	-600.0000060375432	-600.0000000375433	8983	10.51	3502
ex5_2_2_case3	9	6	T	-750.0000075432918	-750.0000000432917	1846	2.25	1163
ex5_2_4	7	6	T	-450.0000045467874	-450.0000000467874	9437	8.11	8813
ex5_4_2	8	6	T	7512.230069378222	7512.230144500523	3704	3.66	1628

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex6_1_1	9	6	F	-0.021229887574120403	-0.01757940731198984	2200000	3703.80	545957
ex6_1_2	5	3	T	-0.032463794582323124	-0.03246378458232312	138	0.37	62
ex6_1_3	13	9	F	-0.44200050561088316	-0.3243488184307259	1600000	3628.10	546058
ex6_1_4	7	4	T	-0.29454129783452365	-0.29454128783452365	2727	3.02	1160
ex6_2_5	10	3	F	-5831.2075381299055	-70.60034485943328	1999999	1952.08	1999999
ex6_2_6	4	1	T	-2.612527082284922E-6	-2.6025270822849222E-6	1538312	1270.75	130464
ex6_2_7	10	3	F	-60.2516157772796	-0.14532645307029823	1999999	3725.52	1999999
ex6_2_8	4	1	T	-0.02700635884306208	-0.027006348843062077	442948	384.56	29620
ex6_2_9	5	2	T	-0.03406619411329543	-0.03406618411329543	334963	355.23	24751
ex6_2_10	7	3	F	-3.1600327229806137	-3.0519642956859942	4333634	2978.80	1999999
ex6_2_11	4	1	T	-2.6823951679630263E-6	-2.6723951679630264E-6	135635	118.99	12380
ex6_2_12	5	2	T	0.2891947384735326	0.2891947484735326	92057	81.32	6785
ex6_2_13	7	3	F	-0.23526029049254815	-0.21620943687698135	4170289	2415.73	1999999
ex6_2_14	5	2	T	-0.6953579446244851	-0.695357934624485	99282	97.22	8185
ex7_2_1	7	14	T	1227.2259651736892	1227.225977445949	10756	17.43	6478
ex7_2_2	6	5	T	-0.3888114445298021	-0.3888114345298021	583	0.84	450
ex7_2_3	8	6	F	6895.3125	$+\infty$	4700000	3600.61	140997
ex7_2_5	5	6	T	10122.493113234434	10122.493214459366	322	0.57	107
ex7_2_6	3	1	T	-83.24972976977953	-83.24972893728224	2083	1.72	340
ex7_2_10	11	9	T	0.09999999000000001	0.1	37	0.28	37
ex7_3_1	4	7	T	0.3417395427959286	0.3417395527959286	1506	1.72	168
ex7_3_2	4	7	T	1.0898639586247951	1.089863969523435	82	0.31	30
ex7_3_3	5	8	T	0.8175290381865713	0.8175290481865713	204	0.51	132
ex7_3_4	12	17	F	0.0E+0	$+\infty$	3600000	3681.48	1979029
ex7_3_5	13	15	F	0.0E+0	$+\infty$	3072081	3378.27	1999999
ex7_3_6	17	17	T	9.9999999E+299	$+\infty$	1	0.20	0

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex8_1_7	5	5	T	0.02931082178569189	0.029310831785691892	1107	1.38	446
ex8_1_8	6	5	T	-0.3888114445298021	-0.3888114345298021	583	0.85	450
ex9_2_1	11	9	T	16.99999983	17.0	64	0.24	3
ex9_2_2	11	11	T	99.99969382468747	99.99969482468441	589806	331.08	98305
ex9_2_3	17	15	T	-1.0E-8	0.0E+0	35	0.21	5
ex9_2_4	9	7	T	0.4999999904656613	0.5000000004656613	49	0.23	4
ex9_2_5	9	7	T	4.999999957789511	5.00000000778951	133	0.30	29
ex9_2_6	17	12	F	-1.341126559073465	-1.0	3200000	3676.93	1239563
ex9_2_7	11	9	T	16.99999983	17.0	64	0.25	3
ex14_1_1	3	4	T	-1.64749944854621E-8	-6.474994485462099E-9	185	0.78	15
ex14_1_2	6	9	T	-1.4899778617507546E-8	-4.8997786175075454E-9	23777	19.45	7063
ex14_1_3	3	4	T	-1.0039153797418883E-8	-3.9153797418882765E-11	66	0.25	14
ex14_1_5	6	6	T	-1.922099021532675E-8	-9.22099021532675E-9	665	1.03	52
ex14_1_6	9	15	T	-1.2515662153715131E-8	-2.515662153715131E-9	1484	3.12	380
ex14_1_7	10	17	F	-3.680062693554289	2042.9013648706106	3900000	3705.58	341793
ex14_1_8	3	4	T	-1.3559221983878366E-8	-3.559221983878366E-9	73	0.22	26
ex14_1_9	2	2	T	-1.8273710585355795E-8	-8.273710585355795E-9	220	0.37	48
ex14_2_1	5	7	T	-1.0E-8	0.0E+0	33362	38.10	3176
ex14_2_2	4	5	T	-1.0E-8	0.0E+0	1437	1.26	67
ex14_2_3	6	9	T	-1.0E-8	0.0E+0	112721	166.45	9649
ex14_2_4	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	72572	81.95	3548
ex14_2_5	4	5	T	-1.0E-8	0.0E+0	2621	2.27	141
ex14_2_6	5	7	T	-1.0E-8	0.0E+0	152460	210.87	45538
ex14_2_7	6	9	F	0.0E+0	0.10913936421275139	1900000	3773.64	632962
ex14_2_8	4	5	T	-1.0E-8	0.0E+0	18853	18.89	461
ex14_2_9	4	5	T	-1.0E-8	0.0E+0	12789	13.83	1496
Moyenne lorsque '**T**'		62				61151.37	53.83	6924.97

A.6 Résultats de $IBBA+rART_{AF2}+CP$

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
hs071	4	2	T	17.0140171933527	17.014017363492872	809	1.04	735
ex2_1_1	5	1	T	-17.00000003961484	-16.99999986961484	151	0.27	20
ex2_1_2	6	2	T	-213.0000018338394	-212.99999970383942	105	0.24	105
ex2_1_3	13	9	T	-15.000000014376148	-14.99999986437615	266	0.42	260
ex2_1_4	6	5	T	-11.0000000753548	-10.9999999653548	245	0.39	223
ex2_1_5	10	11	T	-268.01463415946074	-268.0146314793144	268	0.49	199
ex2_1_6	10	5	T	-39.00000004727329	-38.99999965727329	285	0.50	40
ex2_1_7	20	10	T	-4150.4101750826184	-4150.410133578517	1574	3.53	1142
ex2_1_8	24	10	T	15638.999860875556	15639.000017265556	3359	6.78	3359
ex2_1_9	10	1	T	-0.3750000101909464	-0.3750000001909464	60007	52.97	25762
ex2_1_10	20	10	T	49318.01747194881	49318.017965128994	640	1.11	640
ex3_1_1	8	6	T	7049.247950036507	7049.248020528988	55492	53.73	18625
ex3_1_2	5	6	T	-30665.53897927148	-30665.538672616094	111	0.24	44
ex3_1_3	6	6	T	-310.00000129459334	-309.9999981945934	182	0.28	133
ex3_1_4	3	3	T	-4.000000006472387	-3.9999999664723873	183	0.28	23
ex4_1_8	2	1	T	-16.73889332479729	-16.738893157408355	127	0.21	61
ex4_1_9	2	2	T	-5.5080133221501874	-5.508013267070055	157	0.29	15
ex5_2_2_case1	9	6	T	-400.00000385302564	-399.99999985302565	5353	6.38	2728
ex5_2_2_case2	9	6	T	-600.0000058134244	-599.9999998134244	9307	11.81	3767
ex5_2_2_case3	9	6	T	-750.0000073890417	-749.9999998890416	2211	2.68	1509
ex5_2_4	7	6	T	-450.00000438318346	-449.9999998831835	9914	8.47	9126
ex5_4_2	8	6	T	7512.230069378556	7512.230144500858	137661	59.64	7938

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex6_1_1	9	6	F	-0.021413450741576245	$+\infty$	2000000	3622.57	476057
ex6_1_2	5	3	T	-0.032463794582323124	-0.03246378458232312	138	0.32	62
ex6_1_3	13	9	F	-0.4420004536027947	$+\infty$	1600000	3816.64	571731
ex6_1_4	7	4	T	-0.29454129783452365	-0.29454128783452365	2748	3.24	1160
ex6_2_5	10	3	F	-5831.2075381299055	$+\infty$	1999999	1925.24	1999999
ex6_2_6	4	1	T	-2.6125251699812733E-6	-2.6025251699812734E-6	1543008	1394.78	131721
ex6_2_7	10	3	F	-60.57264234790793	$+\infty$	1900000	3621.72	1900000
ex6_2_8	4	1	T	-0.02700635884306208	-0.027006348843062077	446971	413.38	29621
ex6_2_9	5	2	T	-0.03406619411329543	-0.03406618411329543	337766	368.95	24751
ex6_2_10	7	3	F	-3.1600429496935396	-3.0519497528589663	4333400	3024.25	1999999
ex6_2_11	4	1	T	-2.682392735172946E-6	-2.672392735172946E-6	135891	125.55	12380
ex6_2_12	5	2	T	0.2891947384735326	0.2891947484735326	92837	86.22	6785
ex6_2_13	7	3	F	-0.2352609662316439	-0.21620841615131775	4170149	2492.90	1999999
ex6_2_14	5	2	T	-0.6953579385812835	-0.6953579285812834	134351	145.79	19856
ex7_2_1	7	14	T	1227.2260630076812	1227.226075279942	12418	21.24	7505
ex7_2_2	6	5	T	-0.38881144934638917	-0.3888114393463892	558	0.76	493
ex7_2_3	8	6	F	6881.25	$+\infty$	4500000	3624.55	128939
ex7_2_5	5	6	T	10122.493217568921	10122.493318793855	361	0.50	142
ex7_2_6	3	1	T	-83.24972967399799	-83.2497288415007	2566	2.08	504
ex7_2_10	11	9	T	0.09999999595450439	0.10000000595450438	1417	1.32	1262
ex7_3_1	4	7	T	0.34173955182411774	0.34173956182411774	1536	1.92	705
ex7_3_2	4	7	T	1.0898639601992741	1.0898639710979138	141	0.29	55
ex7_3_3	5	8	T	0.8175290406218059	0.8175290506218059	373	0.59	206
ex7_3_4	12	17	F	0.0E+0	$+\infty$	3275442	3673.86	1999999
ex7_3_5	13	15	F	-7.152557373046875E-7	$+\infty$	2228770	3417.17	1999999
ex7_3_6	17	17	T	9.9999999E+299	$+\infty$	1	0.15	0

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex8_1_7	5	5	T	0.02931082245928056	0.02931083245928056	1439	1.53	1158
ex8_1_8	6	5	T	-0.38881144934638917	-0.3888114393463892	558	0.78	493
ex9_2_1	11	9	T	16.99999983	17.0	64	0.26	3
ex9_2_2	11	11	F	99.99993642173118	$+\infty$	5902793	3451.84	1999999
ex9_2_3	17	15	T	-1.0E-8	0.0E+0	156	0.43	11
ex9_2_4	9	7	T	0.4999999904656613	0.5000000004656613	49	0.22	5
ex9_2_5	9	7	T	4.999999976410212	5.000000026410213	136	0.39	56
ex9_2_6	17	12	F	-1.3411265590734657	56.3203125	2476882	3446.02	1999999
ex9_2_7	11	9	T	16.99999983	17.0	64	0.26	3
ex14_1_1	3	4	T	-1.9078677963007207E-8	-9.078677963007207E-9	300	0.46	15
ex14_1_2	6	9	T	-7.88613338736261E-9	2.1138666126373895E-9	23976	20.69	7226
ex14_1_3	3	4	T	-2.049408845572937E-9	7.950591154427063E-9	94	0.26	22
ex14_1_5	6	6	T	-3.075417549157069E-9	6.924582450842931E-9	2996	3.61	399
ex14_1_6	9	15	T	-1.511833642110571E-9	8.48816635788943E-9	2266	5.38	380
ex14_1_7	10	17	F	-3.680062693554289	3234.9943010627066	3800000	3651.51	341202
ex14_1_8	3	4	T	-5.930893681546904E-9	4.0691063184530965E-9	94	0.26	26
ex14_1_9	2	2	T	-5.218028628659333E-9	4.781971371340667E-9	821	0.75	115
ex14_2_1	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	138055	147.07	4110
ex14_2_2	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	2212	1.84	114
ex14_2_3	6	9	T	-1.3263826201159647E-9	8.673617379884035E-9	119849	178.00	11470
ex14_2_4	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	51533	63.20	12735
ex14_2_5	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	2581	2.35	170
ex14_2_6	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	171690	231.39	36147
ex14_2_7	6	9	F	0.0E+0	0.07275957614183426	1800000	3608.86	604408
ex14_2_8	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	23403	23.19	276
ex14_2_9	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	16211	17.45	1706
Moyenne lorsque '**T**'		61				58361.23	57.03	6398.39

A.7 Résultats de $IBBA+rART_{fAF2}+CP$

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
hs071	4	2	T	17.014017195440946	17.014017365581118	1103	1.12	735
ex2_1_1	5	1	T	-17.00000003961484	-16.99999986961484	151	0.12	20
ex2_1_2	6	2	T	-213.0000018338394	-212.99999970383942	105	0.08	105
ex2_1_3	13	9	T	-15.000000014376148	-14.99999986437615	266	0.23	260
ex2_1_4	6	5	T	-11.000000027671081	-10.999999917671083	224	0.18	108
ex2_1_5	10	11	T	-268.01463416730075	-268.0146314871544	263	0.33	203
ex2_1_6	10	5	T	-39.00000004727329	-38.99999965727329	285	0.26	40
ex2_1_7	20	10	T	-4150.410174740917	-4150.410133236815	1488	3.90	1106
ex2_1_8	24	10	T	15638.999863250661	15639.000019640662	2546	5.81	2546
ex2_1_9	10	1	T	-0.37500000091958324	-0.37499999091958324	89257	86.43	33559
ex2_1_10	20	10	T	49318.01747045439	49318.01796363457	636	1.09	636
ex3_1_1	8	6	T	7049.2479500417384	7049.248020534219	45337	50.30	18386
ex3_1_2	5	6	T	-30665.53897927148	-30665.538672616094	111	0.10	44
ex3_1_3	6	6	T	-310.00000129459334	-309.9999981945934	182	0.14	133
ex3_1_4	3	3	T	-4.000000038137355	-3.999999998137355	179	0.15	23
ex4_1_8	2	1	T	-16.73889332479729	-16.738893157408355	127	0.07	61
ex4_1_9	2	2	T	-5.5080133221501874	-5.508013267070055	157	0.13	15
ex5_2_2_case1	9	6	T	-400.0000037442501	-399.9999997442501	5187	6.17	2600
ex5_2_2_case2	9	6	T	-600.0000058134244	-599.9999998134244	9120	11.54	3643
ex5_2_2_case3	9	6	T	-750.000007302094	-749.999999802094	2165	2.52	1473
ex5_2_4	7	6	T	-450.00000459286422	-450.00000009286424	9787	8.46	9032
ex5_4_2	8	6	T	7512.230069378843	7512.230144501144	27599	15.15	3499

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex6_1_1	9	6	T	-0.020198320866919236	-0.020198310866919234	1725033	3161.74	420354
ex6_1_2	5	3	T	-0.032463794582323124	-0.03246378458232312	108	0.11	46
ex6_1_3	13	9	F	-0.40455211619891306	1.0E+300	1500000	3740.83	547390
ex6_1_4	7	4	T	-0.29454129783452365	-0.29454128783452365	1622	2.87	692
ex6_2_5	10	3	F	-5475.418935750781	1.0E+300	1999999	1911.03	1999999
ex6_2_6	4	1	T	-2.6123794235108254E-6	-2.6023794235108255E-6	921910	927.98	81151
ex6_2_7	10	3	F	-35.91155114465631	1.0E+300	1600000	3663.47	1600000
ex6_2_8	4	1	T	-0.027006358842591345	-0.027006348842591343	265318	269.43	18364
ex6_2_9	5	2	T	-0.03406619411329543	-0.03406618411329543	204240	250.83	15583
ex6_2_10	7	3	F	-3.164280437898766	-3.0519497528589663	4200000	3657.38	1877821
ex6_2_11	4	1	T	-2.682392735172946E-6	-2.672392735172946E-6	83481	81.33	6996
ex6_2_12	5	2	T	0.2891947384735326	0.2891947484735326	58231	58.10	4096
ex6_2_13	7	3	F	-0.23412940257870296	-0.21620841615131775	4287127	3027.83	1999999
ex6_2_14	5	2	T	-0.6953579385812835	-0.6953579285812834	95945	116.37	15973
ex7_2_1	7	14	T	1227.2260665512837	1227.2260788235446	8415	15.06	5284
ex7_2_2	6	5	T	-0.38881144934638917	-0.3888114393463892	531	0.79	469
ex7_2_3	8	6	T	7049.248019148852	7049.248089641333	2486037	2797.47	107519
ex7_2_5	5	6	T	10122.493217568921	10122.493318793855	186	3.51	85
ex7_2_6	3	1	T	-83.24972967399799	-83.2497288415007	1319	1.20	319
ex7_2_10	11	9	T	0.09999999595450439	0.10000000595450438	1417	1.19	1262
ex7_3_1	4	7	T	0.34173955182411774	0.34173956182411774	1536	2.02	705
ex7_3_2	4	7	T	1.0898639601992741	1.0898639710979138	141	0.32	55
ex7_3_3	5	8	T	0.8175290395376036	0.8175290495376036	496	0.77	241
ex7_3_4	12	17	F	-1.7285346984863281E-6	1.0E+300	2700000	3622.91	1963184
ex7_3_5	13	15	F	-8.344650268554687E-6	1.0E+300	2200000	3717.38	1970372
ex7_3_6	17	17	T	9.9999999E+299	1.0E+300	1	0.28	0

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex8_1_7	5	5	T	0.02931082245928056	0.02931083245928056	1437	1.63	1158
ex8_1_8	6	5	T	-0.38881144934638917	-0.3888114393463892	531	0.81	469
ex9_2_1	11	9	T	16.99999983	17.0	64	0.28	3
ex9_2_2	11	11	F	99.99993642172952	1.0E+300	5902793	3500.72	1999999
ex9_2_3	17	15	T	-2.869632592373819E-9	7.130367407626181E-9	161	0.43	42
ex9_2_4	9	7	T	0.4999999904656613	0.5000000004656613	49	0.28	5
ex9_2_5	9	7	T	4.999999976410212	5.000000026410213	159	0.41	33
ex9_2_6	17	12	F	-1.3411265590737384	56.3203125	2430385	3443.14	1999999
ex9_2_7	11	9	T	16.99999983	17.0	64	1.33	3
ex14_1_1	3	4	T	-1.907867963007207E-8	-9.078677963007207E-9	300	0.52	15
ex14_1_2	6	9	T	-7.88613338736261E-9	2.1138666126373895E-9	23976	22.44	7226
ex14_1_3	3	4	T	-2.049408845572937E-9	7.950591154427063E-9	91	0.31	20
ex14_1_5	6	6	T	-3.075417549157069E-9	6.924582450842931E-9	2218	2.94	355
ex14_1_6	9	15	T	-7.652145135845236E-10	9.234785486415476E-9	2577	6.57	380
ex14_1_7	10	17	F	-3.680062693554289	3234.9943010627066	3500000	3695.28	331905
ex14_1_8	3	4	T	-1.2065084208750782E-9	8.793491579124922E-9	77	0.35	23
ex14_1_9	2	2	T	-7.013255592648307E-9	2.9867444073516935E-9	242	0.41	36
ex14_2_1	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	16829	22.60	2559
ex14_2_2	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	1018	1.15	163
ex14_2_3	6	9	T	-3.4947869650869736E-9	6.505213034913027E-9	47842	87.84	9706
ex14_2_4	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	29936	47.16	8758
ex14_2_5	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	2041	2.15	188
ex14_2_6	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	74672	121.59	24002
ex14_2_7	6	9	F	0.0E+0	0.10913936421275139	1500000	3623.26	559386
ex14_2_8	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	10107	11.91	403
ex14_2_9	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	6582	8.53	603
Moyenne lorsque '**T**'			63			99574.84	130.59	12913.83

A.8 Résultats de $IBBA+rART_{sAF2}+CP$

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
hs071	4	2	T	17.014017195440946	17.014017365581118	1103	2.06	735
ex2_1_1	5	1	T	-17.00000003961484	-16.99999986961484	151	0.44	20
ex2_1_2	6	2	T	-213.0000018338394	-212.99999970383942	105	0.39	105
ex2_1_3	13	9	T	-15.000000014376148	-14.99999986437615	266	1.28	260
ex2_1_4	6	5	T	-11.0000000753548	-10.9999999653548	231	0.49	104
ex2_1_5	10	11	T	-268.01463416730075	-268.0146314871544	263	1.76	203
ex2_1_6	10	5	T	-39.00000004727329	-38.99999965727329	285	1.72	40
ex2_1_7	20	10	T	-4150.410175477704	-4150.410133973603	1530	50.13	1123
ex2_1_8	24	10	T	15638.999860875556	15639.000017265556	3507	147.02	3507
ex2_1_9	10	1	T	-0.37500000161808966	-0.37499999161808966	83673	423.15	22710
ex2_1_10	20	10	T	49318.01747045439	49318.01796363457	636	16.74	636
ex3_1_1	8	6	T	7049.247950652987	7049.248021145468	47180	99.52	18338
ex3_1_2	5	6	T	-30665.53897927148	-30665.538672616094	111	0.40	44
ex3_1_3	6	6	T	-310.00000129459334	-309.9999981945934	182	0.58	133
ex3_1_4	3	3	T	-4.000000038137355	-3.999999998137355	182	0.42	23
ex4_1_8	2	1	T	-16.73889332479729	-16.738893157408355	127	0.09	61
ex4_1_9	2	2	T	-5.5080133221501874	-5.508013267070055	157	0.41	15
ex5_2_2_case1	9	6	T	-400.000003819192	-399.999999819192	5254	15.26	2659
ex5_2_2_case2	9	6	T	-600.0000058134244	-599.9999998134244	9195	26.55	3681
ex5_2_2_case3	9	6	T	-750.0000073170097	-749.9999998170097	2115	6.29	1413
ex5_2_4	7	6	T	-450.0000043958287	-449.9999998958287	9842	18.38	9093
ex5_4_2	8	6	T	7512.230069380636	7512.230144502938	4241	8.73	1939

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex6_1_1	9	6	F	-0.021625487294624257	1.0E+300	1000000	3791.47	256048
ex6_1_2	5	3	T	-0.032463794582323124	-0.03246378458232312	108	0.36	46
ex6_1_3	13	9	F	-0.4717201934175885	1.0E+300	500000	3940.63	191232
ex6_1_4	7	4	T	-0.29454129783452365	-0.29454128783452365	1622	4.57	692
ex6_2_5	10	3	F	-8633.66991725223	1.0E+300	200000	3657.01	200000
ex6_2_6	4	1	F	-2.6175216915872927E-6	-2.602397202677853E-6	900000	3622.02	81779
ex6_2_7	10	3	F	-77.71862682596038	1.0E+300	200000	5759.67	200000
ex6_2_8	4	1	T	-0.027006358842591345	-0.027006348842591343	265316	1071.23	18364
ex6_2_9	5	2	T	-0.03406619411329543	-0.03406618411329543	204272	1291.05	15583
ex6_2_10	7	3	F	-4.1338020606289944	-3.0519497528589663	400000	4014.02	220464
ex6_2_11	4	1	T	-2.682392735172946E-6	-2.672392735172946E-6	83488	285.68	6996
ex6_2_12	5	2	T	0.2891947384735326	0.2891947484735326	58231	259.33	4096
ex6_2_13	7	3	F	-0.36942144483001915	-0.2162011652309288	500000	3634.90	287452
ex6_2_14	5	2	T	-0.6953579385812835	-0.6953579285812834	95166	456.06	16714
ex7_2_1	7	14	T	1227.2260665512837	1227.2260788235446	8416	50.63	5287
ex7_2_2	6	5	T	-0.38881144934638917	-0.3888114393463892	531	1.19	469
ex7_2_3	8	6	F	6932.707427031278	1.0E+300	1300000	3665.27	51417
ex7_2_5	5	6	T	10122.493217568921	10122.493318793855	186	0.67	85
ex7_2_6	3	1	T	-83.24972967399799	-83.2497288415007	1319	1.54	319
ex7_2_10	11	9	T	0.09999999595450439	0.10000000595450438	1417	4.02	1262
ex7_3_1	4	7	T	0.34173955182411774	0.34173956182411774	1536	6.83	705
ex7_3_2	4	7	T	1.0898639601992741	1.0898639710979138	141	0.48	55
ex7_3_3	5	8	T	0.8175290395376036	0.8175290495376036	496	1.19	241
ex7_3_4	12	17	F	-2.6226043701171875E-6	1.0E+300	400000	4008.88	270698
ex7_3_5	13	15	F	-2.6226043701171875E-6	1.0E+300	200000	4495.04	177277
ex7_3_6	17	17	T	9.9999999E+299	1.0E+300	1	0.21	0

Nom	N	M	ok ?	Minorant	Majorant	Iter	t (s)	max $\|\mathcal{L}\|$
ex8_1_7	5	5	T	0.02931082245928056	0.02931083245928056	1434	4.69	1156
ex8_1_8	6	5	T	-0.38881144934638917	-0.3888114393463892	531	1.23	469
ex9_2_1	11	9	T	16.99999983	17.0	64	0.21	3
ex9_2_2	11	11	F	99.99987284350001	1.0E+300	2900000	3701.73	987157
ex9_2_3	17	15	T	-2.869632592373819E-9	7.130367407626181E-9	161	1.53	42
ex9_2_4	9	7	T	0.4999999904656613	0.5000000004656613	49	0.11	5
ex9_2_5	9	7	T	4.999999976410212	5.000000026410213	159	0.37	33
ex9_2_6	17	12	F	-1.5098876953125206	56.3203125	600000	3778.58	487094
ex9_2_7	11	9	T	16.99999983	17.0	64	1.18	3
ex14_1_1	3	4	T	-1.9078677963007207E-8	-9.078677963007207E-9	300	0.60	15
ex14_1_2	6	9	T	-7.88613338736261E-9	2.1138666126373895E-9	23976	134.43	7226
ex14_1_3	3	4	T	-2.049408845572937E-9	7.950591154427063E-9	91	0.13	20
ex14_1_5	6	6	T	-3.075417549157069E-9	6.924582450842931E-9	2973	5.70	356
ex14_1_6	9	15	T	-3.4183256272596766E-9	6.581674372740323 5E-9	2338	25.42	380
ex14_1_7	10	17	F	-66.54433862555936	3234.9943010627066	500000	4017.37	3850
ex14_1_8	3	4	T	-1.2065084208750782E-9	8.793491579124922E-9	77	1.17	23
ex14_1_9	2	2	T	-1.4308548406939394E-8	-4.308548406939394E-9	223	0.24	36
ex14_2_1	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	16804	78.12	2559
ex14_2_2	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	1018	1.97	163
ex14_2_3	6	9	T	-1.3263826201159647E-9	8.673617379884035E-9	47896	440.32	9706
ex14_2_4	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	29946	287.35	8758
ex14_2_5	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	2041	6.56	188
ex14_2_6	5	7	T	-1.3263826201159647E-9	8.673617379884035E-9	74672	643.02	24002
ex14_2_7	6	9	F	0.0E+0	0.21827872842550277	300000	5257.87	112508
ex14_2_8	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	10106	45.02	403
ex14_2_9	4	5	T	-1.3263826201159647E-9	8.673617379884035E-9	6581	34.81	603
Moyenne lorsque '**T**'			60			18568.1	99.52	3231.75

Annexe B

Détails des résultats numériques de la section 5.2.2

Cette annexe présente les détails des solutions trouvées lors des tests effectués dans la section 5.2.2 pour résoudre le problème de dimensionnement d'un moteur électrique (PB_m). Les algorithmes utilisés pour résoudre le problème de dimensionnement sont les suivants :

1. ***IBBA :*** *IBBA* (algorithme 2) incluant la technique de propagation des contraintes (algorithme 3).
2. ***IBBA+ART :*** *IBBA* (algorithme 2) incluant la technique de propagation des contraintes (algorithme 3) et la technique de reformulation affine (algorithme 8) utilisant l'arithmétique affine AF2.
3. ***IBBA+rART :*** *IBBA* (algorithme 2) incluant la technique de propagation des contraintes (algorithme 3) et la technique de reformulation affine robuste (algorithme 9) utilisant l'arithmétique affine robuste rAF2 (section 2.5.2).
4. ***IBBA+MART :*** *IBBA* (algorithme 2) incluant la technique de propagation des contraintes (algorithme 3) et la technique de reformulation affine mixte (section 3.3) utilisant l'arithmétique affine AF2.

Le tableau B.1 indique les solutions obtenues par les différents algorithmes pour résoudre le problème de minimisation de la masse du moteur M_a, le tableau B.2 ceux de la minimisation du volume global V_g du moteur et le tableau B.3 ceux de la minimisation de la fonction multicritère $MULTI$ (voir équation 5.2).

TABLE B.1 – Solutions de la minimisation de la masse du moteur.

Variable	Lettre	*IBBA*	*IBBA+ART*	*IBBA+rART*	*IBBA+MART*
x_1	D	0.139829	0.138553	0.138553	0.138644
x_2	L	0.050976	0.050000	0.050000	0.049999
x_3	e	0.001	0.001	0.001	0.009999
x_4 (10^{-3})	l_a	4.124759	4.000487	4.000487	3.999999
x_5 (10^{-3})	E	6.021652	6.591952	6.591952	6.574728
x_6 (10^{-3})	C	4.374279	4.326032	4.326032	4.323215
x_7	β	0.85	0.85	0.85	0.85
x_8	j	5789062.5	5614608.7	5614608.7	5612960.8
x_9 (10^{-3})	a	4.198273	4.000303	4.000303	4.015384
x_{10} (10^{-3})	d	4.559433	4.636540	4.636540	4.628564
x_{11}	Θ_J	130	130	130	130
x_{12}	Θ_c	55	55	55	55
x_{13}	Θ_e	50	50	50	50
z_1	p	8	8	8	8
z_2	q	3	3	3	3
z_3	m	1	1	1	1
b_1	b_r	0	0	0	0
b_2	b_e	1	1	1	1
b_3	b_f	1	1	1	1
k_1	σ_m	Moderne	Moderne	Moderne	Moderne
k_2	σ_{mt}	Poudre	Poudre	Poudre	Poudre
Masse	M_a (kg)	2.3177	2.2449	2.2449	2.2456
Volume	V_g $(10^{-3}m^3)$	1.0099	0.9705	0.9705	0.9715
Multi-obj	$MULTI$	2.123032	2.048033	2.048033	2.049451

TABLE B.2 – Solutions de la minimisation du volume du moteur.

Variable	Lettre	*IBBA*	*IBBA+ART*	*IBBA+rART*	*IBBA+MART*
x_1	D	-	0.132744	0.132796	0.132132
x_2	L	-	0.050006	0.050018	0.050036
x_3	e	-	0.001	0.001	0.001
x_4 (10^{-3})	l_a	-	5.401855	5.363183	5.669335
x_5 (10^{-3})	E	-	7.183137	7.184922	7.156380
x_6 (10^{-3})	C	-	4.129635	4.127335	4.141796
x_7	β	-	0.85	0.85	0.85
x_8	j	-	5596435.5	5596435.5	5587646.4
x_9 (10^{-3})	a	-	4.004872	4.005232	4.006466
x_{10} (10^{-3})	d	-	5.387111	5.390475	5.341719
x_{11}	Θ_J	-	130	130	130
x_{12}	Θ_c	-	55	55	55
x_{13}	Θ_e	-	50	50	50
z_1	p	-	7	7	7
z_2	q	-	3	3	3
z_3	m	-	1	1	1
b_1	b_r	-	0	0	0
b_2	b_e	-	1	1	1
b_3	b_f	-	1	1	1
k_1	σ_m	-	Moderne	Moderne	Moderne
k_2	σ_{mt}	-	Tôle	Tôle	Tôle
Masse	M_a (kg)	-	2.6964	2.6910	2.7336
Volume	V_g $(10^{-3}m^3)$	-	0.9291	0.9289	0.9290
Multi-obj	$MULTI$	-	2.204982	2.202424	2.221510

TABLE B.3 – Solutions de la minimisation du multi-critère du moteur.

Variable	Lettre	*IBBA*	*IBBA+ART*	*IBBA+rART*	*IBBA+MART*
x_1	D	-	0.138553	0.138553	0.138515
x_2	L	-	0.050000	0.050000	0.050003
x_3	e	-	0.001	0.001	0.001
x_4 (10^{-3})	l_a	-	4.000243	4.000243	4.001949
x_5 (10^{-3})	E	-	6.592203	6.591700	6.588834
x_6 (10^{-3})	C	-	4.325788	4.325788	4.325545
x_7	β	-	0.85	0.85	0.85
x_8	j	-	5614608.7	5614608.7	5613647.4
x_9 (10^{-3})	a	-	4.000303	4.000303	4.002237
x_{10} (10^{-3})	d	-	4.636540	4.636573	4.632355
x_{11}	Θ_J	-	130	130	130
x_{12}	Θ_c	-	55	55	55
x_{13}	Θ_e	-	50	50	50
z_1	p	-	8	8	8
z_2	q	-	3	3	3
z_3	m	-	1	1	1
b_1	b_r	-	0	0	0
b_2	b_e	-	1	1	1
b_3	b_f	-	1	1	1
k_1	σ_m	-	Moderne	Moderne	Moderne
k_2	σ_{mt}	-	Poudre	Poudre	Poudre
Masse	M_a (kg)	-	2.2448	2.2448	2.2446
Volume	V_g $(10^{-3}m^3)$	-	0.9705	0.9705	0.9701
Multi-obj	$MULTI$	-	2.047986	2.047970	2.047486